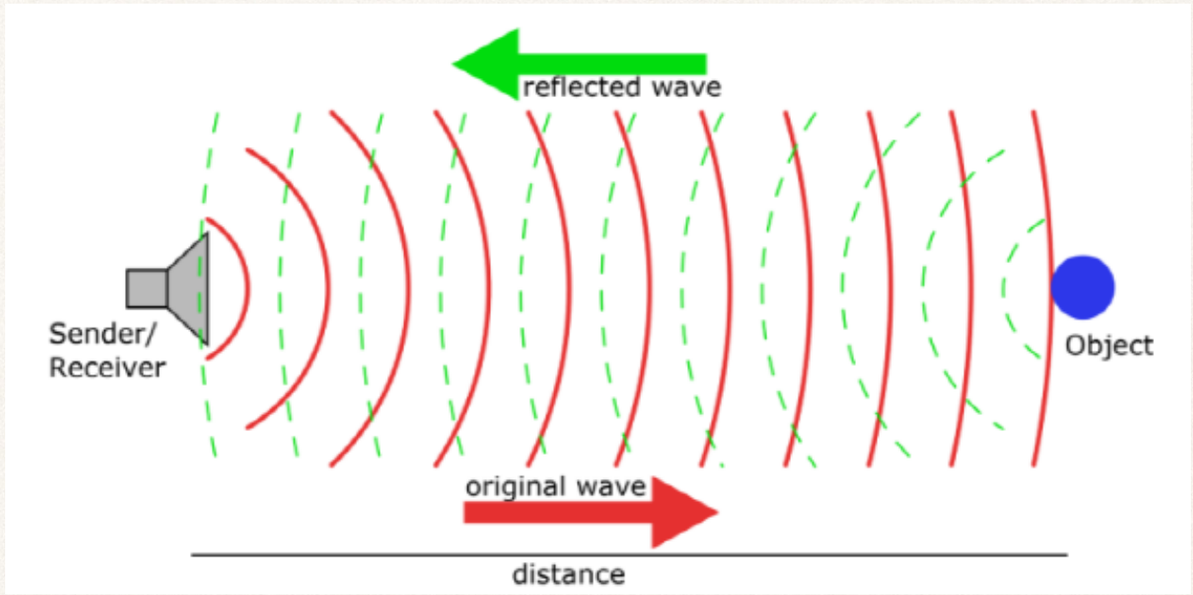
Point-GNN

Graph Neural Network for 3D Object Detection in a Point Cloud, CVPR 2020

Weijing Shi, Ragunathan Rajkumar

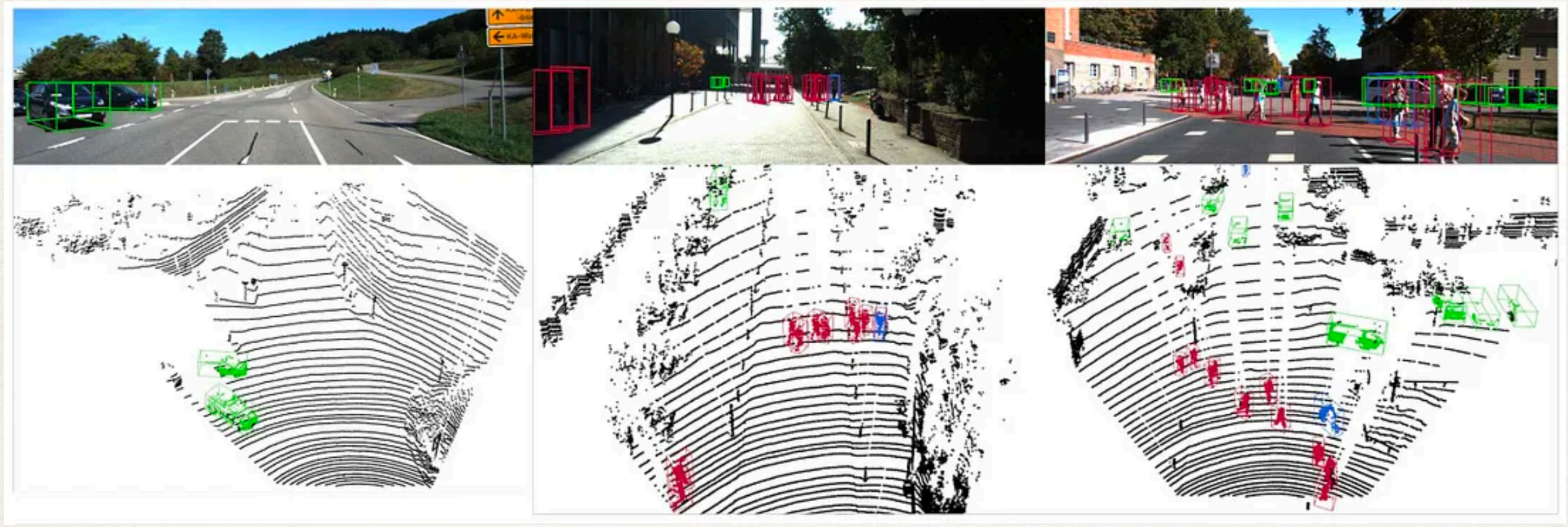
TADA S2 : Network Analysis

Introduction



- **Point Cloud**
  - Digital 3D representation of a physical object or space.
  - Made of millions of individual measurement points - each one with x, y, z coordinate.
  - Created by 3D scanning methods - LiDAR or photogrammetry that measures the distance between the scanner and nearby objects.
    - Available from iPhone 12 Pro +, Samsung Galaxy S10 5G

Introduction



- Point-GNN detects a 3D object in a point cloud with a graph neural network
  - The model predicts the category and shape of the object that each vertex in the graph belongs to.
    - Auto-registration mechanism to reduce translation variance
    - Box merging and scoring operation to combine detections from multiple vertices

Introduction

- Previous Works
  - Convolution is commonly used for object detection, however it requires a regular grid as an input. A point cloud is sparser and uneven to be put on a regular grid.
  - Neural networks with unordered set of points (point set representation) as input were used to extract point cloud features. However repeated grouping and sampling on a large point cloud can be computationally costly.
- Current Work
  - Graph neural network is used by encoding the point cloud - points as graph vertices and the edges connected with neighborhood points that lie within a fixed radius.
    - Does not use regular grids (image or a voxel) for representation, does not group or sample the points repeatedly - graph is constructed only once.

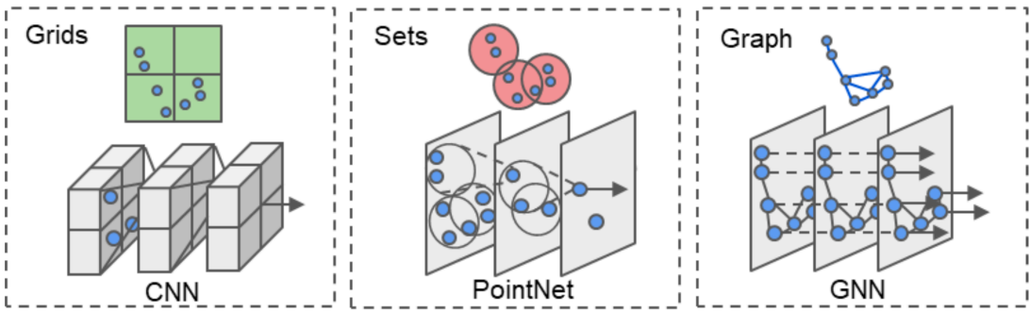


Figure 1. Three point cloud representations and their common processing methods.

Overall Architecture

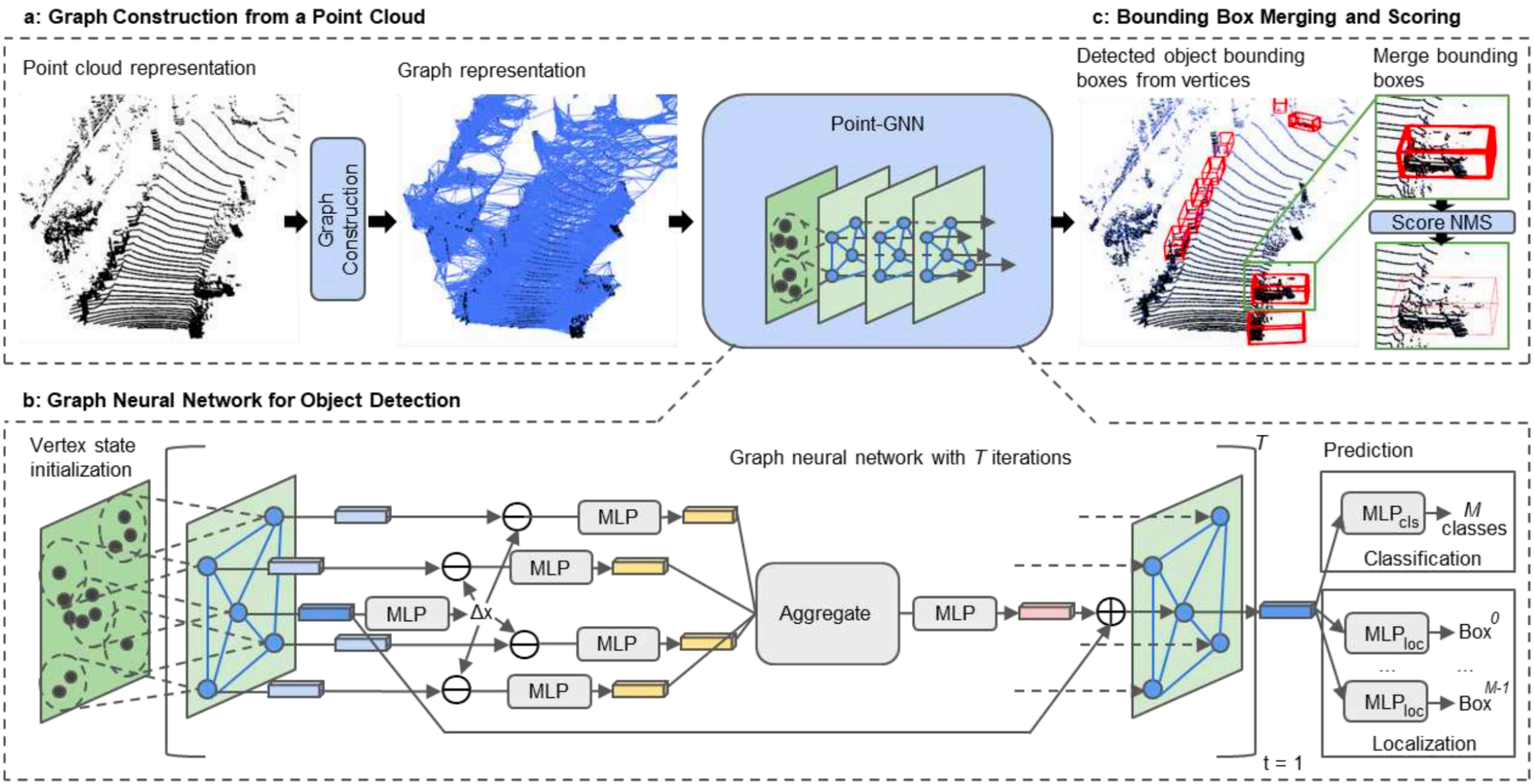
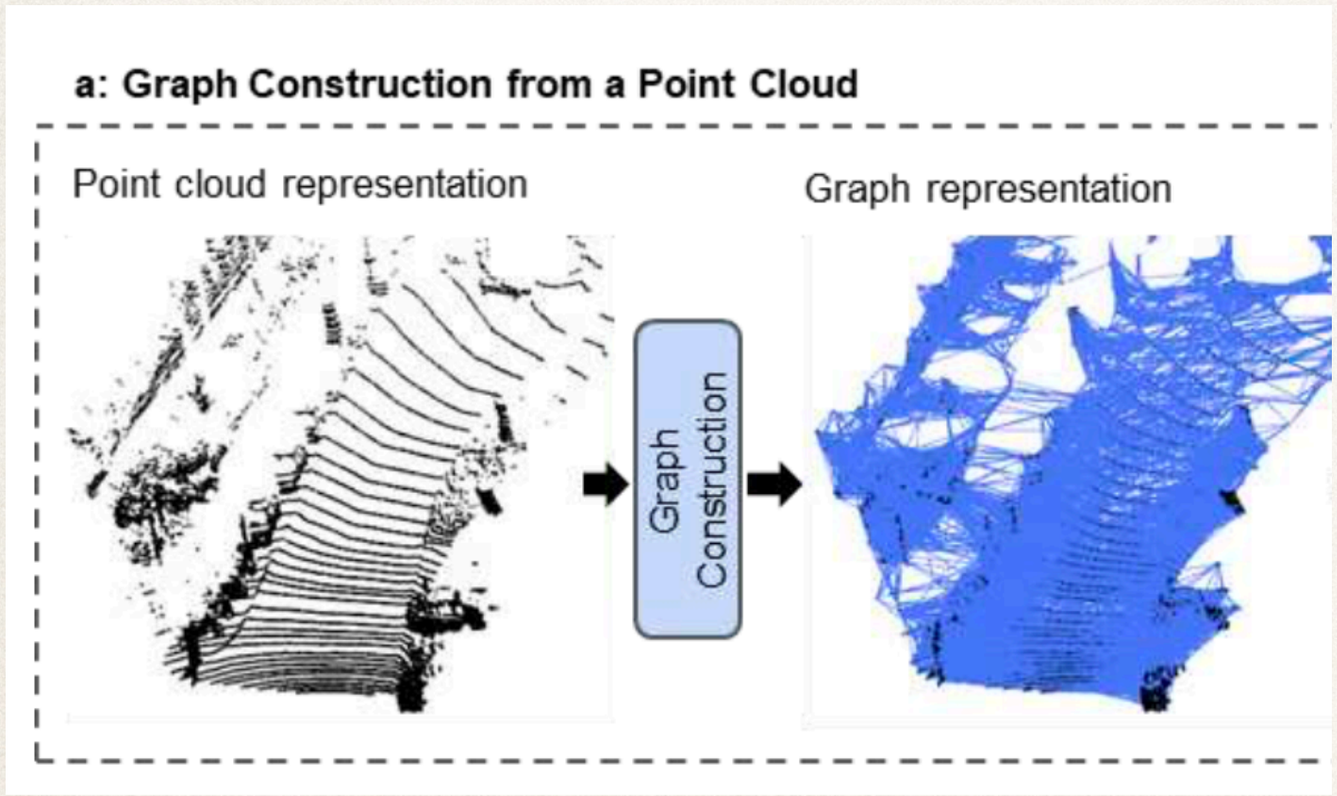


Figure 2. The architecture of the proposed approach. It has three main components: (a) graph construction from a point cloud, (b) a graph neural network for object detection, and (c) bounding box merging and scoring.

Graph Construction from a Point Cloud

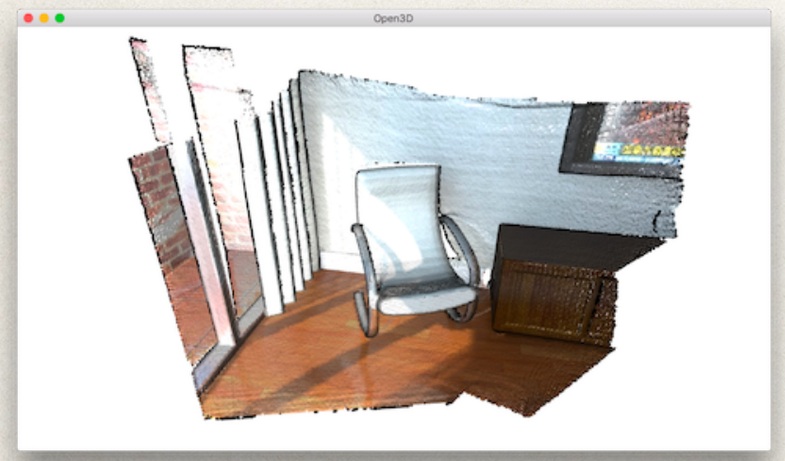


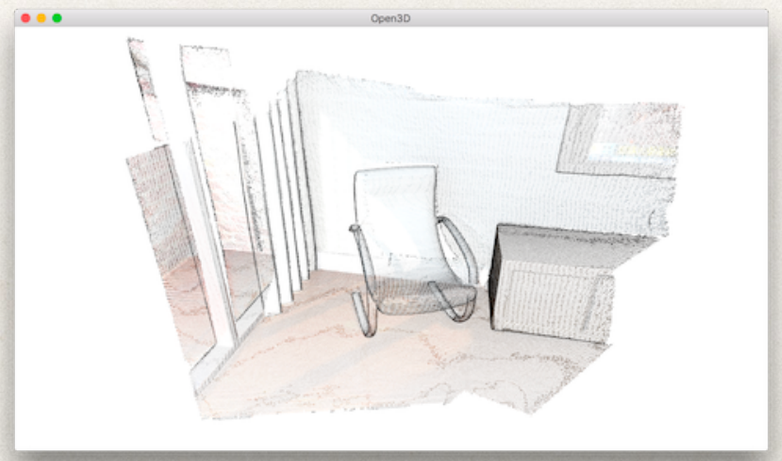
$$E = \{(p_i, p_j) \mid \|x_i - x_j\|_2 < r\}$$

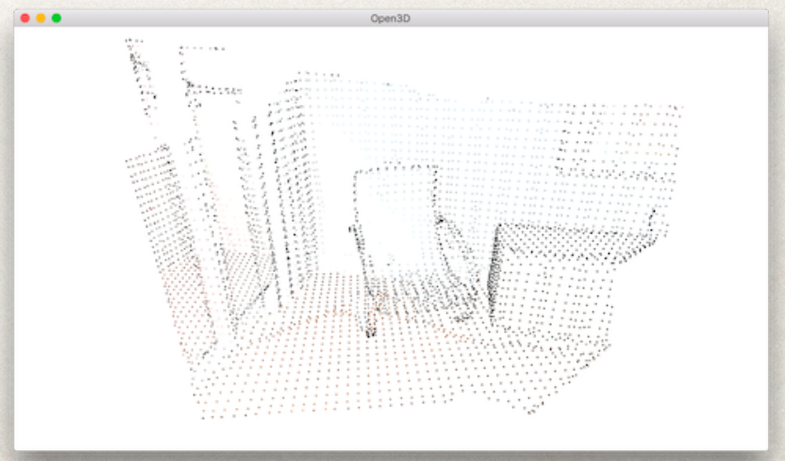
- Given a point cloud P, a graph G = (P, E) is constructed by connecting a point to its neighbors within a fixed radius r.
  - Finding point pairs that are within a cut-off distance - O(cN)
    - c : max number of neighbors within the radius
    - N : points in the point cloud

Graph Construction from a Point Cloud

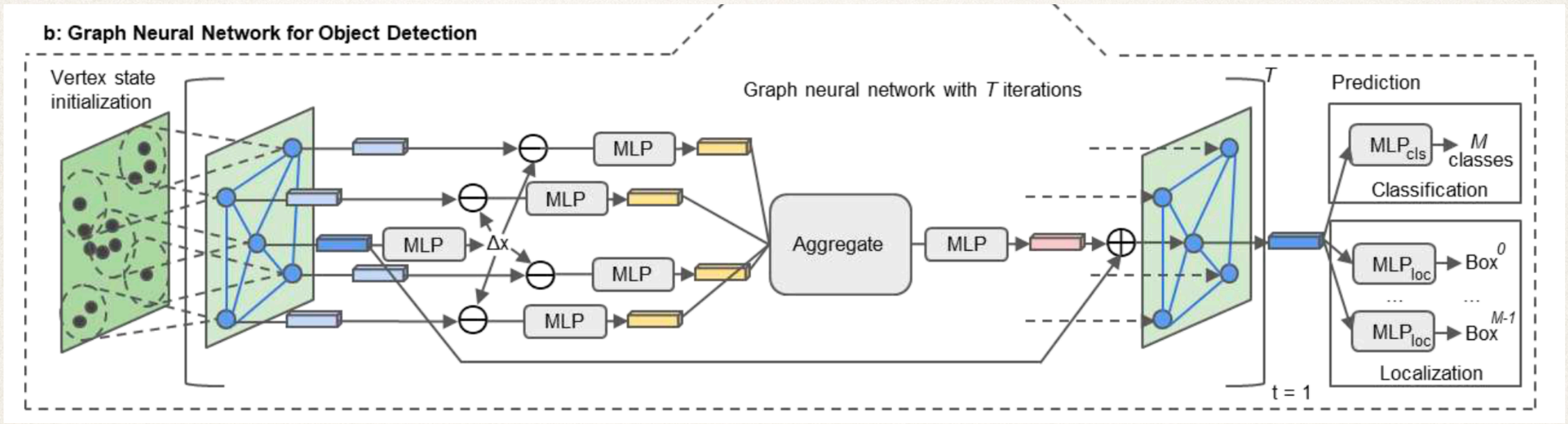
- Voxel Downsampling
  - Dividing the 3D space that contains the point cloud into a regular grid of cubic voxels. Each voxel represents a small region of space, and all the points that fall within a voxel are replaced by a single representative point.
  - The number of points in the point cloud is reduced, which can make it more computationally tractable to process the data.
  - PointGNN uses voxel downsampling as a preprocessing step to reduce the point cloud density before constructing the graph representation and applying graph convolutional operations. This allows PointGNN to handle large-scale point cloud data efficiently while still retaining important spatial information.
  - It must be noted that the **voxels here are only used to reduce the density of a point cloud and they are not used as the representation of the point cloud.**







Graph Neural Network for Object Detection (Point-GNN)



- For each vertex, the central vertex and its relative coordinates are embedded in the MLP to extract features, then each feature vector is aggregated by Max function (which selects the maximum value of each feature across all the raw points within the radius of r)
- This step allows the Point-GNN algorithm to extract relevant features from the raw point cloud data and represent them in a form suitable for input to the subsequent layers of the GNN

Graph Neural Network for Object Detection (Point-GNN)

Auto-Registration

- In the context of lidar point cloud data, small movements can be caused by the motion of the lidar sensor or the object being scanned. For example, if the lidar sensor is mounted on a moving platform such as a car, the sensor's motion can cause small translations in the point cloud data.
- If the object being scanned is moving or rotating, it can cause small movements in the point cloud data.

- Point-GNN algorithm proposes 'Auto-registration' to align the neighbors' coordinates by their structural features to reduce the translation variance and improve the accuracy of object detection.

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

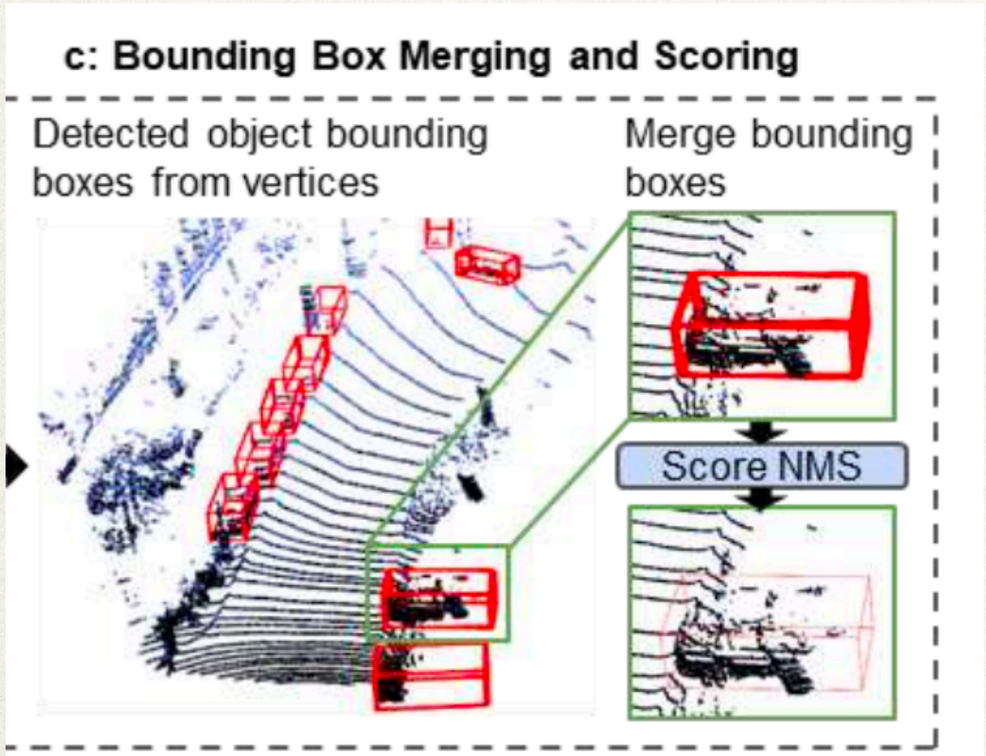
Δx_i^t is the coordination offset for the vertices to register coordinates.

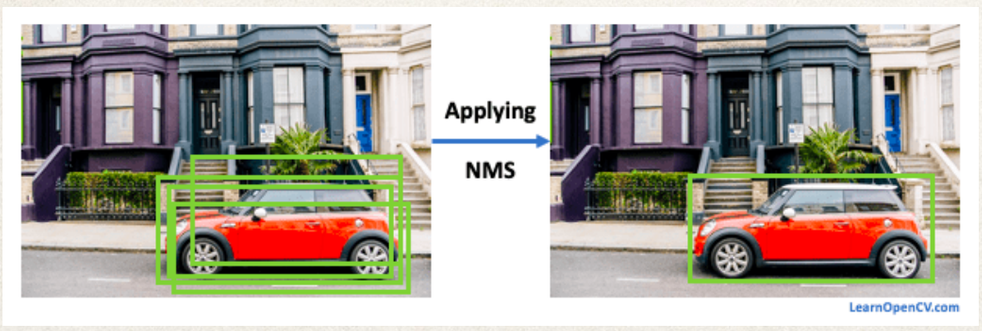
Graph Neural Network for Object Detection (Point-GNN)

Loss

- For each vertex in the point cloud, the algorithm computes a probability distribution {pc1 , ..., pcM } for M object classes, including the background class.
- If a vertex is within the bounding box of an object, the algorithm assigns the corresponding object class to the vertex. If a vertex is not inside any bounding boxes, the algorithm assigns the background class to it.
- To train the classification branch, the algorithm uses the average cross-entropy loss as the classification loss. (L1 regularization)

Bounding Box Scoring and Merging





- The neural network may output multiple bounding boxes for the same object, which need to be merged into one and assigned a confidence score. Non-maximum suppression (NMS) is a widely used technique for merging bounding boxes.
- The common practice is to select the bounding box with the highest classification score and suppress the other overlapping boxes. However, the classification score may not always reflect the quality of the localization. For example, a partially occluded object may have a strong clue indicating its type, but lack enough shape information. Therefore, using only the classification score to select the best bounding box may not be sufficient.
- To calculate the merged box, the median position and size of the overlapping bounding boxes are taken into account. Additionally, a confidence score is computed based on the sum of the classification scores, weighted by the Intersection-of-Union (IoU) factor and an occlusion factor.
  - The occlusion factor represents the occupied volume ratio of the bounding box. To calculate the occlusion factor, the length, width, and height of the box are determined, along with the unit vectors that indicate their directions. The coordinates of the point are also taken into account. The occlusion factor is then calculated based on this information.

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

Experiments

- Dataset : KITTI object detection benchmark
  - 7481 training samples and 7518 testing samples
    - 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.

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.

Point-GNN outperformed fusion-based algorithms in all categories except for Pedestrian detection. Paper mentions that one likely reason why Pedestrian detection is not as good as that for Car and Cyclist is that the vertices are not dense enough to achieve more accurate bounding boxes.

Experiments

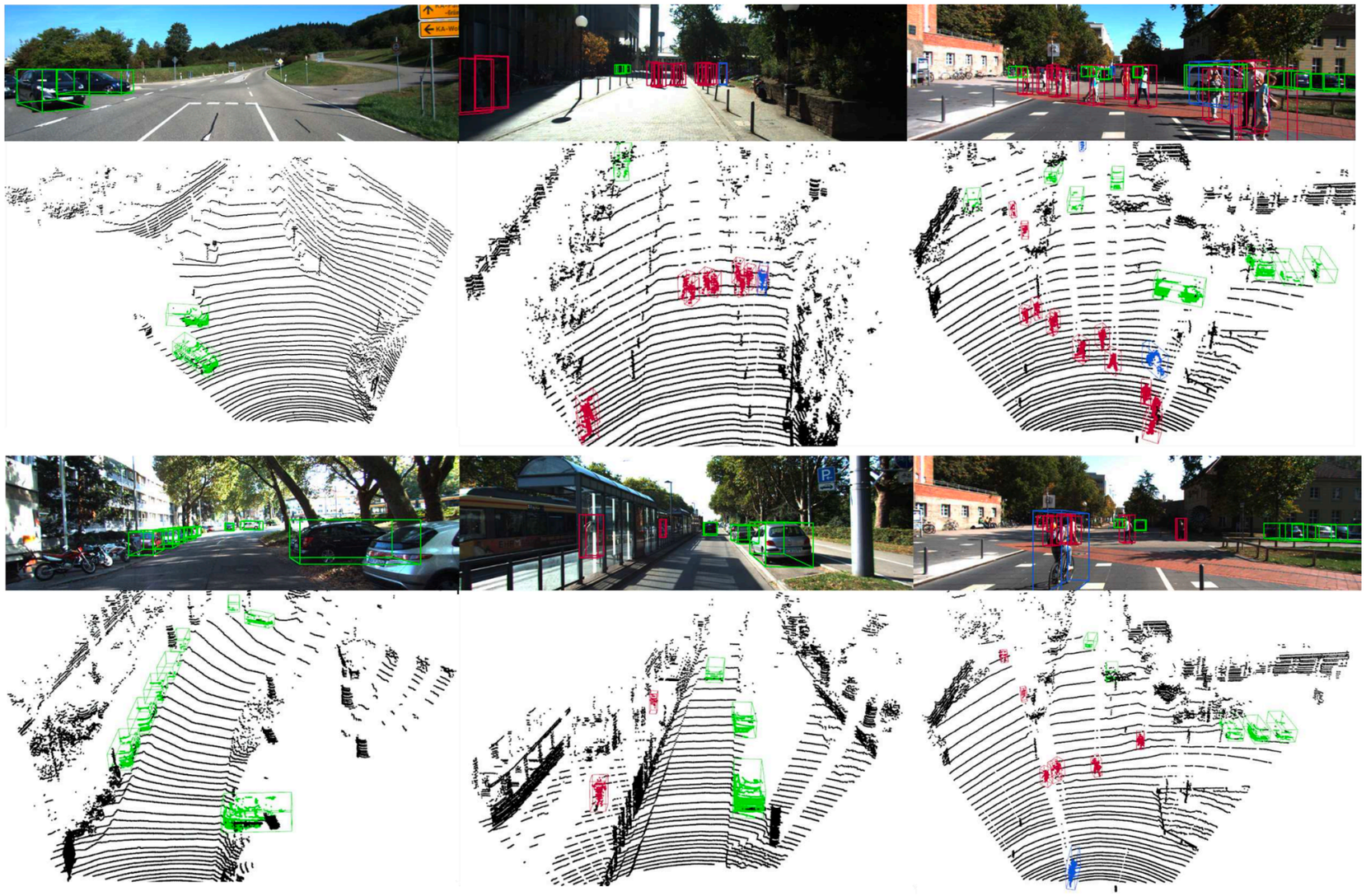


Figure 3. Qualitative results on the KITTI *test* dataset using Point-GNN. We show the predicted 3D bounding box of Cars (green), Pedestrians (red) and Cyclists (blue) on both the image and the point cloud. Best viewed in color.

References

- Shi, W., Rajkumar, R. (2020). Point-GNN : Graph Neural Networks for 3D Object Detection in a Point Cloud. CVPR 2020
- 2d3d.ai (2021, Oct 4). Graph Neural Networks for Point Cloud Processing. Youtube. https://www.youtube.com/watch?v=jrATl7Y48Hg&t=2865s&ab_channel=2d3d.ai