

VoxelNet¹, and some others

Adam Schmidt

February 10, 2021

¹Zhou and Tuzel, 2018.

Table of Contents

Introduction

VoxelNet

Alternatives

Opinion Disclaimer

I do not like voxels.

What We Want

- 3D bounding boxes given LiDAR data

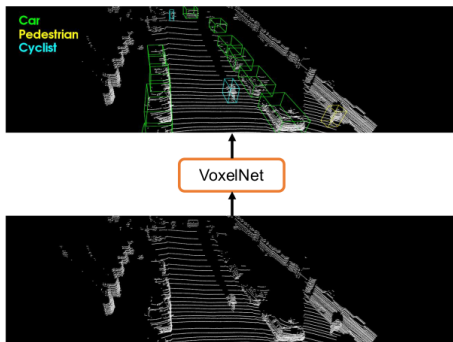


Figure 1. VoxelNet directly operates on the raw point cloud (no need for feature engineering) and produces the 3D detection results using a single end-to-end trainable network.

2

²Zhou and Tuzel, 2018.

Steps

- Make voxels

Steps

- Make voxels
- Process voxels

Steps

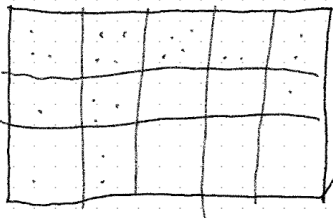
- Make voxels
- Process voxels
- Convert to 2D

Steps

- Make voxels
- Process voxels
- Convert to 2D
- Propose regions

Make Voxels

- Pool all points features in voxel, and append to features



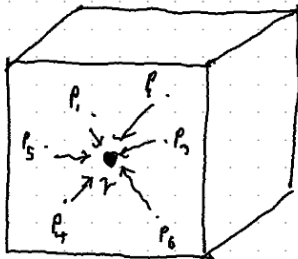
Break space into voxels

- limit to T points per voxel

- Random sample if there are more than T

For each Voxel

- find centroid
 r

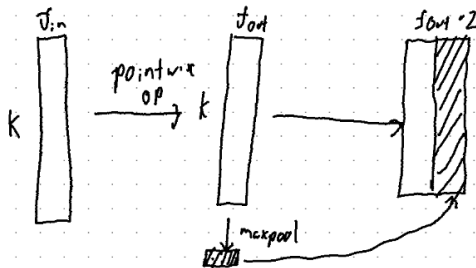


For each Voxel

$$f_{in}(p_i) = (\underbrace{x_i, y_i, z_i}_{\text{position}}, \underbrace{r_i}_{\text{redistance}}, \underbrace{x_i - r_x, y_i - r_y, z_i - r_z}_{\text{offset from centroid}})$$

For each Voxel

- Pool all points features in voxel, and append to features



Convolutional Mid-layers

Set of 3D convolutions that reduce over a couple layers until $D = 2$, effectively flattening the data into an image.

Loss

$$\begin{aligned} L &= \alpha \frac{1}{N_{\text{pos}}} \sum_i L_{\text{cls}}(p_i^{\text{pos}}, 1) + \beta \frac{1}{N_{\text{neg}}} \sum_j L_{\text{cls}}(p_j^{\text{neg}}, 0) \\ &+ \frac{1}{N_{\text{pos}}} \sum_i L_{\text{reg}}(\mathbf{u}_i, \mathbf{u}_i^*) \end{aligned} \quad (2)$$

Results

| Method | Modality | Car | | | Pedestrian | | | Cyclist | | |
|--------------------|------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| | | Easy | Moderate | Hard | Easy | Moderate | Hard | Easy | Moderate | Hard |
| Mono3D [3] | Mono | 5.22 | 5.19 | 4.13 | N/A | N/A | N/A | N/A | N/A | N/A |
| 3DOP [4] | Stereo | 12.63 | 9.49 | 7.59 | N/A | N/A | N/A | N/A | N/A | N/A |
| VeloFCN [22] | LiDAR | 40.14 | 32.08 | 30.47 | N/A | N/A | N/A | N/A | N/A | N/A |
| MV (BV+FV) [5] | LiDAR | 86.18 | 77.32 | 76.33 | N/A | N/A | N/A | N/A | N/A | N/A |
| MV (BV+FV+RGB) [5] | LiDAR+Mono | 86.55 | 78.10 | 76.67 | N/A | N/A | N/A | N/A | N/A | N/A |
| HC-baseline | LiDAR | 88.26 | 78.42 | 77.66 | 58.96 | 53.79 | 51.47 | 63.63 | 42.75 | 41.06 |
| VoxelNet | LiDAR | 89.60 | 84.81 | 78.57 | 65.95 | 61.05 | 56.98 | 74.41 | 52.18 | 50.49 |

Table 1. Performance comparison in bird's eye view detection: average precision (in %) on KITTI validation set.

| Method | Modality | Car | | | Pedestrian | | | Cyclist | | |
|--------------------|------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| | | Easy | Moderate | Hard | Easy | Moderate | Hard | Easy | Moderate | Hard |
| Mono3D [3] | Mono | 2.53 | 2.31 | 2.31 | N/A | N/A | N/A | N/A | N/A | N/A |
| 3DOP [4] | Stereo | 6.55 | 5.07 | 4.10 | N/A | N/A | N/A | N/A | N/A | N/A |
| VeloFCN [22] | LiDAR | 15.20 | 13.66 | 15.98 | N/A | N/A | N/A | N/A | N/A | N/A |
| MV (BV+FV) [5] | LiDAR | 71.19 | 56.60 | 55.30 | N/A | N/A | N/A | N/A | N/A | N/A |
| MV (BV+FV+RGB) [5] | LiDAR+Mono | 71.29 | 62.68 | 56.56 | N/A | N/A | N/A | N/A | N/A | N/A |
| HC-baseline | LiDAR | 71.73 | 59.75 | 55.69 | 43.95 | 40.18 | 37.48 | 55.35 | 36.07 | 34.15 |
| VoxelNet | LiDAR | 81.97 | 65.46 | 62.85 | 57.86 | 53.42 | 48.87 | 67.17 | 47.65 | 45.11 |

Table 2. Performance comparison in 3D detection: average precision (in %) on KITTI validation set.

4

⁴Ren et al., 2015.

Whole Architecture

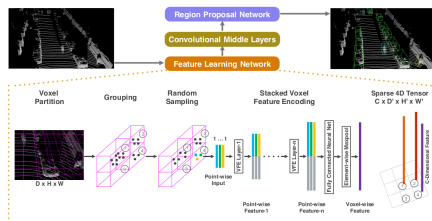


Figure 2. VoxNet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers processes the 4D tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

Questions

- Points cannot communicate outside their voxels

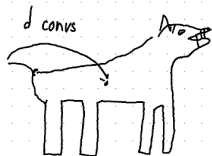
Questions

- Points cannot communicate outside their voxels
- Should we be using a 2D to detect in 3D?

Questions

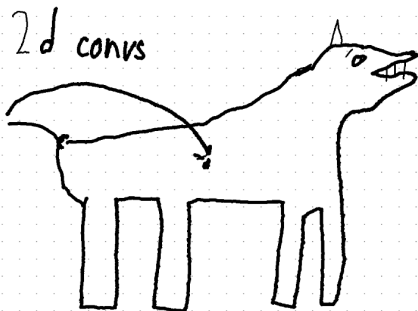
- Points cannot communicate outside their voxels
- Should we be using a 2D to detect in 3D?
- Feature receptive field

Field



what does
network do when
already at 'center'
- fixed point

what if we haven't
reached center?



Requires voting, NMS

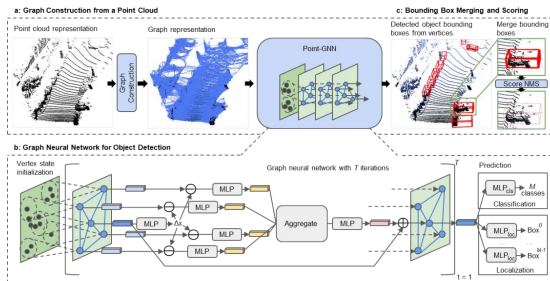


Figure 2. The architecture of the proposed approach. It has three main components: (a) graph construction from a point cloud, (b) a graph neural network for object detection, and (c) bounding box merging and scoring. ⁵

⁵Shi, Ragunathan, and Rajkumar, 2020.

Deep Hough Transform

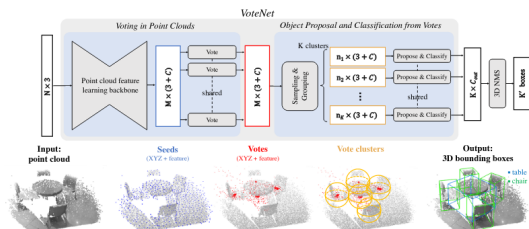


Figure 2. Illustration of the VoteNet architecture for 3D object detection in point clouds. Given an input point cloud of N points with XYZ coordinates, a backbone network (implemented with PointNet++ [36] layers) subsamples and learns deep features on the points and outputs a subset of M points but extended by C -dim features. This subset of points are considered as seed points. Each seed independently generates a vote through a voting module. Then the votes are grouped into clusters and processed by the proposal module to generate the final proposals. The classified and NMSed proposals become the final 3D bounding boxes output. Image best viewed in color.

6

⁶Qi et al., 2019.

⁷Shi, Ragunathan, and Rajkumar, 2020.

Deep Hough Transform

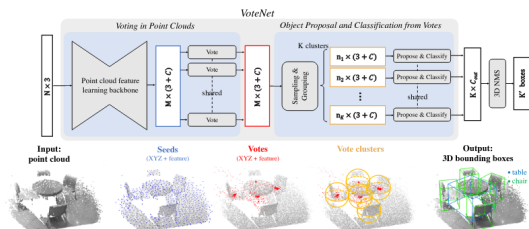


Figure 2. Illustration of the VoteNet architecture for 3D object detection in point clouds. Given an input point cloud of N points with XYZ coordinates, a backbone network (implemented with PointNet++ [36] layers) subsamples and learns deep features on the points and outputs a subset of M points but extended by C -dim features. This subset of points are considered as seed points. Each seed independently generates a vote through a voting module. Then the votes are grouped into clusters and processed by the proposal module to generate the final proposals. The classified and NMSed proposals become the final 3D bounding boxes output. Image best viewed in color.

6

PointRCNN⁷ less pretty, but also used

⁶Qi et al., 2019.

⁷Shi, Ragunathan, and Rajkumar, 2020.

Pointformer

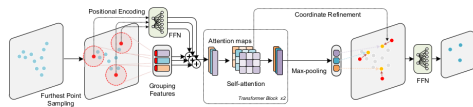
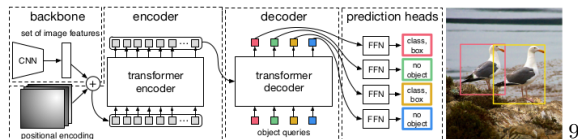


Figure 3. Illustration of the Local Transformer. Input points are first down-sampled by FPS and generate local regions by ball query. Transformer block takes point features and coordinates as input and generate aggregated features for the local region. To further adjust the centroid points, attention maps from the last Transformer layer are adopted for coordinate refinement. As a result, points are pushed closer to the object centers instead of surfaces.

8




Honorable Mention

Transformer, positional encoding, and Non-Maximum Suppression-free!



⁹Carion et al., 2020.

References I

-  Carion, Nicolas et al. (May 28, 2020). *End-to-End Object Detection with Transformers*. Version 1. arXiv: 2005.12872 [cs]. URL: <http://arxiv.org/abs/2005.12872> (visited on 09/30/2020).
-  Pan, Xuran et al. (Dec. 21, 2020). *3D Object Detection with Pointformer*. arXiv: 2012.11409 [cs]. URL: <http://arxiv.org/abs/2012.11409> (visited on 01/05/2021).
-  Qi, Charles R. et al. (Aug. 22, 2019). *Deep Hough Voting for 3D Object Detection in Point Clouds*. arXiv: 1904.09664 [cs]. URL: <http://arxiv.org/abs/1904.09664> (visited on 02/09/2021).

References II



Ren, Shaoqing et al. (June 4, 2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv: 1506.01497 [cs]. URL: <http://arxiv.org/abs/1506.01497> (visited on 03/18/2019).



Shi, Weijing, Raguathan, and Rajkumar (Mar. 2, 2020). *Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud*. arXiv: 2003.01251 [cs]. URL: <http://arxiv.org/abs/2003.01251> (visited on 03/09/2020).

References III



Zhou, Yin and Oncel Tuzel (2018). “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4490–4499. URL: https://openaccess.thecvf.com/content_cvpr_2018/html/Zhou_VoxelNet_End-to-End_Learning_CVPR_2018_paper.html (visited on 01/27/2021).