3D Point Processing

James Hays
Recap – Structured Output from Deep Networks

A lot of machine learning tools, such as convolutional networks, don’t naturally handle tasks with arbitrary numbers of outputs. These are a few clever methods, typical of the literature as a whole, to work around this.
Outline

• How do we measure 3D points?
• How do we make decisions about point clouds?
  • PointNet – orderless point processing
  • VoxelNet – voxel-based point processing
  • PointPillars – bird’s eye view point processing
    • Exploiting Visibility for 3D Object Detection
    • LaserNet – range image point processing
• PseudoLidar – Bird’s eye view depth map processing
Kinect V1 and V2

Infrared images of Kinect V1 structured light pattern and Kinect V2 time of flight pattern. Credit “Lightweight Algorithms for Depth Sensor Equipped Embedded Devices” by Henry Zhong
Lidar overview
Lidar overview

Source: Waymo Open Dataset
Outline

• What is lidar?

• How do we make decisions about point clouds?
  • PointNet – orderless point processing
  • VoxelNet – voxel-based point processing
  • PointPillars – bird’s eye view point processing
    • Exploiting Visibility for 3D Object Detection
    • LaserNet – range image point processing

• PseudoLidar – Bird’s eye view depth map processing
PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

Charles R. Qi* Hao Su* Kaichun Mo Leonidas J. Guibas
Big Data + Deep Representation Learning

Robot Perception

Augmented Reality

Shape Design

source: Scott J Grunewald

source: Google Tango

source: solidsolutions

Need for 3D Deep Learning!
3D Representations

Point Cloud  Mesh  Volumetric  Projected View RGB(D)
Point cloud is close to raw sensor data
3D Representation: Point Cloud

- Point cloud is close to raw sensor data
- Point cloud is canonical

LiDAR → Point Cloud → Mesh
Depth Sensor → Point Cloud → Volumetric

Mesh → Depth Map
**Previous Works**

Most existing point cloud features are **handcrafted** towards specific tasks

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Supports Texture / Color</th>
<th>Local / Global / Regional</th>
<th>Best Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFH</td>
<td>No</td>
<td>L</td>
<td>2.5D Scans (Pseudo single position range images)</td>
</tr>
<tr>
<td>FPFH</td>
<td>No</td>
<td>L</td>
<td>Object detection with basic pose estimation</td>
</tr>
<tr>
<td>VFH</td>
<td>No</td>
<td>G</td>
<td>Object detection with basic pose estimation, detection of partial objects</td>
</tr>
<tr>
<td>CVFH</td>
<td>No</td>
<td>R</td>
<td>Real world 3D-Scans with no mirror effects. RIFT is vulnerable against flipping.</td>
</tr>
<tr>
<td>RIFT</td>
<td>Yes</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

Source: [https://github.com/PointCloudLibrary/pcl/wiki/Overview-and-Comparison-of-Features](https://github.com/PointCloudLibrary/pcl/wiki/Overview-and-Comparison-of-Features)
Point cloud is **converted to other representations** before it’s fed to a deep neural network.
Research Question:

Can we achieve effective **feature learning directly** on point clouds?
Our Work: PointNet

End-to-end learning for **scattered, unordered** point data
Our Work: PointNet

End-to-end learning for **scattered, unordered** point data

**Unified** framework for various tasks

- Object Classification
- Object Part Segmentation
- Semantic Scene Parsing
- ...
Our Work: PointNet

End-to-end learning for \textbf{scattered, unordered} point data

\textbf{Unified} framework for various tasks

- Classification
- Part Segmentation
- Semantic Segmentation
Challenges

**Unordered point set as input**

Model needs to be invariant to $N!$ permutations.

**Invariance under geometric transformations**

Point cloud rotations should not alter classification results.
Challenges

Unordered point set as input

Model needs to be invariant to $N!$ permutations.

Invariance under geometric transformations

Point cloud rotations should not alter classification results.
Unordered Input

Point cloud: $N$ orderless points, each represented by a $D$ dim vector
Unordered Input

Point cloud: N orderless points, each represented by a D dim vector

represents the same set as
Unordered Input

Point cloud: N orderless points, each represented by a D dim vector

Model needs to be invariant to N! permutations
Permutation Invariance: Symmetric Function

\[ f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D \]
Permutation Invariance: Symmetric Function

\[ f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D \]

Examples:

\[ f(x_1, x_2, \ldots, x_n) = \max \{ x_1, x_2, \ldots, x_n \} \]

\[ f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n \]

\[ \ldots \]
Permutation Invariance: Symmetric Function

\[ f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D \]

Examples:

\[ f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\} \]
\[ f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n \]

\[
\ldots
\]

How can we construct a family of symmetric functions by neural networks?
Permutation Invariance: Symmetric Function

Observe:

\[ f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n)) \] is symmetric if \[ g \] is symmetric
Permutation Invariance: Symmetric Function

Observe:

\[ f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n)) \] is symmetric if \( g \) is symmetric
Permutation Invariance: Symmetric Function

Observe:

\[ f(x_1, x_2, \ldots, x_n) = \gamma^o g(h(x_1), \ldots, h(x_n)) \]

is symmetric if \( g \) is symmetric.
Permutation Invariance: Symmetric Function

Observe:

\[ f(x_1, x_2, \ldots, x_n) = \gamma^o g(h(x_1), \ldots, h(x_n)) \] is symmetric if \( g \) is symmetric.

PointNet (vanilla)
Empirically, we use multi-layer perceptron (MLP) and max pooling:
Challenges

Unordered point set as input
Model needs to be invariant to $N!$ permutations.

Invariance under geometric transformations
Point cloud rotations should not alter classification results.
Idea: Data dependent transformation for automatic alignment
The transformation is just matrix multiplication!
T-Net

transform params: \(64 \times 64\)

Matrix Mult.

Input embeddings: \(N \times 64\)

Transformed embeddings: \(N \times 64\)
Embedding Space Alignment

T-Net

transform
params: 64x64

Matrix
Mult.

Input
embeddings:
Nx64

Transformed
embeddings:
Nx64

Regularization:

Transform matrix A 64x64 close to orthogonal:

\[ L_{reg} = \| I - AA^T \|_F^2 \]
PointNet Classification Network

input points\[n \times 3\]
PointNet Classification Network
PointNet Classification Network

The diagram illustrates the architecture of a PointNet classification network. It begins with input points, which go through a transform layer labeled 'input transform'. This is followed by a 'T-Net' layer, then a '3x3 transform' layer, and finally a multiplication layer labeled 'matrix multiply'. The output of the network is a layer labeled 'mlp (64,64)' followed by another transform layer and a classifier output layer labeled 'nx64'.
PointNet Classification Network
PointNet Classification Network
PointNet Classification Network
Extension to PointNet Segmentation Network

Local embedding

Global feature

input transform

mlp (64, 64)

feature transform

mlp (64, 128, 1024)

max pool

1024

global feature

output scores

nx3

nx3

nx64

nx1024

T-Net

3x3 transform

matrix multiply

T-Net

64x64 transform

matrix multiply

nx3

nx64
Extension to PointNet Segmentation Network

input transform
input points

nx3

mlp (64,64)
feature transform
nx64

T-Net
3x3 transform

matrix multiply

nx3

T-Net
64x64 transform

matrix multiply

nx64

mlp (64,128,1024)

nx1024

max pool
1024
global feature

output scores

Local embedding

Point features

nx5

mlp (512,256,128)
output scores

nxm

mlp (128,m)
Results
## Results on Object Classification

<table>
<thead>
<tr>
<th>Model</th>
<th>Input</th>
<th>#views</th>
<th>Accuracy Avg. Class (%)</th>
<th>Accuracy Overall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPH [12]</td>
<td>mesh</td>
<td>-</td>
<td>68.2</td>
<td></td>
</tr>
<tr>
<td><strong>3DShapeNets [29]</strong></td>
<td>volume</td>
<td>1</td>
<td>77.3</td>
<td>84.7</td>
</tr>
<tr>
<td>VoxNet [18]</td>
<td>volume</td>
<td>12</td>
<td>83.0</td>
<td>85.9</td>
</tr>
<tr>
<td>Subvolume [19]</td>
<td>volume</td>
<td>20</td>
<td>86.0</td>
<td><strong>89.2</strong></td>
</tr>
<tr>
<td>LFD [29]</td>
<td>image</td>
<td>10</td>
<td>75.5</td>
<td>-</td>
</tr>
<tr>
<td>MVCNN [24]</td>
<td>image</td>
<td>80</td>
<td><strong>90.1</strong></td>
<td>-</td>
</tr>
<tr>
<td>Ours baseline</td>
<td>point</td>
<td>-</td>
<td>72.6</td>
<td>77.4</td>
</tr>
<tr>
<td>Ours PointNet</td>
<td>point</td>
<td>1</td>
<td>86.2</td>
<td><strong>89.2</strong></td>
</tr>
</tbody>
</table>

Dataset: ModelNet40; Metric: 40-class classification accuracy (%)
Results on Object Part Segmentation
### Results on Object Part Segmentation

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>aero</th>
<th>bag</th>
<th>cap</th>
<th>car</th>
<th>chair</th>
<th>ear</th>
<th>guitar</th>
<th>knife</th>
<th>lamp</th>
<th>laptop</th>
<th>motor</th>
<th>mug</th>
<th>pistol</th>
<th>rocket</th>
<th>skate</th>
<th>table</th>
</tr>
</thead>
<tbody>
<tr>
<td># shapes</td>
<td>2690</td>
<td>76</td>
<td>55</td>
<td>898</td>
<td>3758</td>
<td>69</td>
<td>787</td>
<td>392</td>
<td>1547</td>
<td>451</td>
<td>202</td>
<td>184</td>
<td>283</td>
<td>66</td>
<td>152</td>
<td>5271</td>
<td></td>
</tr>
<tr>
<td>Wu [28]</td>
<td>-</td>
<td>63.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>73.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>74.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>74.8</td>
<td></td>
</tr>
<tr>
<td>Yi [30]</td>
<td>81.4</td>
<td>81.0</td>
<td>78.4</td>
<td>77.7</td>
<td>75.7</td>
<td>87.6</td>
<td>61.9</td>
<td>92.0</td>
<td>85.4</td>
<td>82.5</td>
<td>95.7</td>
<td>70.6</td>
<td>91.9</td>
<td>85.9</td>
<td>53.1</td>
<td>69.8</td>
<td>75.3</td>
</tr>
<tr>
<td>3DCNN</td>
<td>79.4</td>
<td>75.1</td>
<td>72.8</td>
<td>73.3</td>
<td>70.0</td>
<td>87.2</td>
<td>63.5</td>
<td>88.4</td>
<td>79.6</td>
<td>74.4</td>
<td>93.9</td>
<td>58.7</td>
<td>91.8</td>
<td>76.4</td>
<td>51.2</td>
<td>65.3</td>
<td>77.1</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>83.7</strong></td>
<td><strong>83.4</strong></td>
<td><strong>78.7</strong></td>
<td><strong>82.5</strong></td>
<td><strong>74.9</strong></td>
<td><strong>89.6</strong></td>
<td><strong>73.0</strong></td>
<td><strong>91.5</strong></td>
<td><strong>85.9</strong></td>
<td><strong>80.8</strong></td>
<td><strong>95.3</strong></td>
<td><strong>65.2</strong></td>
<td><strong>93.0</strong></td>
<td><strong>81.2</strong></td>
<td><strong>57.9</strong></td>
<td><strong>72.8</strong></td>
<td><strong>80.6</strong></td>
</tr>
</tbody>
</table>

*dataset: ShapeNetPart; metric: mean IoU (%)*
Results on Semantic Scene Parsing

dataset: Stanford 2D-3D-S (Matterport scans)
Robustness to Data Corruption

Dataset: ModelNet40; Metric: 40-class classification accuracy (%)
Robustness to Data Corruption

Less than 2% accuracy drop with 50% missing data

dataset: ModelNet40; metric: 40-class classification accuracy (%)
Robustness to Data Corruption

dataset: ModelNet40; metric: 40-class classification accuracy (%)
Robustness to Data Corruption

Why is PointNet so robust to missing data?

![Graph showing the comparison between PointNet and 3D CNN in terms of accuracy and missing data ratio.](chart.png)
Visualizing Global Point Cloud Features

Which input points are contributing to the global feature? (critical points)
Visualizing Global Point Cloud Features

Original Shape:

Critical Point Sets:
Visualizing Global Point Cloud Features

Which points won’t affect the global feature?
Visualizing Global Point Cloud Features

Original Shape:

Critical Point Set:

Upper bound set:
Visualizing Point Functions
Conclusion

- PointNet is a novel deep neural network that directly consumes point cloud.
- A unified approach to various 3D recognition tasks.
- Rich theoretical analysis and experimental results.

Code & Data Available!
http://stanford.edu/~rqi/pointnet
## Speed and Model Size

<table>
<thead>
<tr>
<th>Model</th>
<th>#params</th>
<th>FLOPs/sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet (vanilla)</td>
<td>0.8M</td>
<td>148M</td>
</tr>
<tr>
<td>PointNet</td>
<td>3.5M</td>
<td>440M</td>
</tr>
<tr>
<td>Subvolume [16]</td>
<td>16.6M</td>
<td>3633M</td>
</tr>
<tr>
<td>MVCNN [20]</td>
<td>60.0M</td>
<td>62057M</td>
</tr>
</tbody>
</table>

Inference time 11.6ms, 25.3ms GTX1080, batch size 8
Permutation Invariance: How about Sorting?

“Sort” the points before feeding them into a network. Unfortunately, there is no canonical order in high dim space.
"Sort" the points before feeding them into a network.

Unfortunately, there is no canonical order in high dim space.

<table>
<thead>
<tr>
<th>Multi-Layer Perceptron (ModelNet shape classification)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unordered Input</td>
<td>12%</td>
</tr>
<tr>
<td>Lexsorted Input</td>
<td>40%</td>
</tr>
<tr>
<td>PointNet (vanilla)</td>
<td>87%</td>
</tr>
</tbody>
</table>
Permutation Invariance: How about RNNs?

Train RNN with permutation augmentation.

However, RNN forgets and order matters.
Permutation Invariance: How about RNNs?

Train RNN with permutation augmentation.

However, RNN forgets and order matters.

<table>
<thead>
<tr>
<th>LSTM Network</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>75%</td>
</tr>
<tr>
<td>PointNet (vanilla)</td>
<td>87%</td>
</tr>
</tbody>
</table>
Outline

• What is lidar?
• How do we make decisions about point clouds?
  • PointNet – orderless point processing
  • VoxelNet – voxel-based point processing
  • PointPillars – bird’s eye view point processing
    • Exploiting Visibility for 3D Object Detection
    • LaserNet – range image point processing
• PseudoLidar – Bird’s eye view depth map processing
VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection
Yin Zhou and Oncel Tuzel. CVPR 2018
VoxelNet Overview
VoxelNet Voxel encoding looks a lot like PointNet
VoxelNet Overview

Region Proposal Network

Convolutional Middle Layers

Feature Learning Network

Voxel Partition

Grouping

Random Sampling

Stacked Voxel Feature Encoding

Sparse 4D Tensor

C x D' x H' x W'
VoxelNet “Convolutional Middle Layers”

• For car detection, divide the world into 10 x 400 x 352 voxels, corresponding to voxels that are 40 cm tall and 20 cm in width/length.
• Uses **3D** convolutions instead of 2D as we’ve seen before.
• The Z / height dimension gets downsampled away after many layers
VoxelNet Overview
VoxelNet Region Proposal Network

\[(x_c^g, y_c^g, z_c^g, l^g, w^g, h^g, \theta^g)\]
VoxelNet qualitative results

Car

Pedestrian

Cyclist
VoxelNet quantitative results

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

Evaluation on KITTI according to 3D IoU
Outline

• What is lidar?
• How do we make decisions about point clouds?
  • PointNet – orderless point processing
  • VoxelNet – voxel-based point processing
  • PointPillars – bird’s eye view point processing
    • Exploiting Visibility for 3D Object Detection
  • LaserNet – range image point processing
• PseudoLidar – Bird’s eye view depth map processing
PointPillars: Fast Encoders for Object Detection from Point Clouds
Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, Oscar Beijbom. CVPR 2019
Outline

• What is lidar?
• How do we make decisions about point clouds?
  • PointNet – orderless point processing
  • VoxelNet – voxel-based point processing
  • PointPillars – bird’s eye view point processing
    • Exploiting Visibility for 3D Object Detection
    • LaserNet – range image point processing
• PseudoLidar – Bird’s eye view depth map processing
What You See Is What You Get
Exploiting Visibility for 3D Object Detection

Peiyun Hu, Jason Ziglar, David Held, Deva Ramanan

Carnegie Mellon University  Argo AI

CVPR 2020
What is a good representation for LiDAR data?

- LiDAR data provides more than just point measurements
- Rays emanating from the sensor to each 3D point must pass through free space
- Representing LiDAR data as \((x, y, z)\)s fundamentally destroys such freespace information
What representations do we have?

Deep Point Representation
PointNet, Qi et al., CVPR’17

Deep Voxel Representation
PointPillars, Lang et al., CVPR’19

Occupancy Voxels
OctoMap, Hornung et al., Autonomous Robots’13

Visibility Augmented Deep Voxels
WYSIWYG, Hu et al., CVPR’20
A Simple Approach to Augment Visibility

Point Cloud → Voxel Grid → Ray-casting → Deep Voxel Representation

Visibility Volume

Deep Voxel Representation

Visibility-augmented Deep Voxel Representation
Visibility-aware LiDAR Synthesis

Naive Object Augmentation
PointPillars, Lang et al., CVPR'19
SECOND, Yan et al., Sensors'18

Visibility-aware Object Augmentation
Should be occluded!
Occluded!
Improve PointPillars by 4.5% in overall mAP

NuScenes Benchmark (test set)

More than 10%
Almost 20%

PointPillars
Ours
Outline

• What is lidar?
• How do we make decisions about point clouds?
  • PointNet – orderless point processing
  • VoxelNet – voxel-based point processing
  • PointPillars – bird’s eye view point processing
    • Exploiting Visibility for 3D Object Detection
  • LaserNet – range image point processing
• PseudoLidar – Bird’s eye view depth map processing
Table 4: BEV Object Detection Performance on KITTI

<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th>Vehicle $AP_{0.7}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy</td>
<td>Moderate</td>
</tr>
<tr>
<td>LaserNet (Ours)</td>
<td>LiDAR</td>
<td>78.25</td>
</tr>
<tr>
<td>PIXOR [28]</td>
<td>LiDAR</td>
<td>81.70</td>
</tr>
<tr>
<td>PIXOR++ [27]</td>
<td>LiDAR</td>
<td><strong>89.38</strong></td>
</tr>
<tr>
<td>VoxelNet [30]</td>
<td>LiDAR</td>
<td>89.35</td>
</tr>
<tr>
<td>MV3D [51]</td>
<td>LiDAR+RGB</td>
<td>86.02</td>
</tr>
<tr>
<td>AVOD [15]</td>
<td>LiDAR+RGB</td>
<td>88.53</td>
</tr>
<tr>
<td>F-PointNet [22]</td>
<td>LiDAR+RGB</td>
<td>88.70</td>
</tr>
<tr>
<td>ContFuse [17]</td>
<td>LiDAR+RGB</td>
<td>88.81</td>
</tr>
</tbody>
</table>

Table 3: Runtime Performance on KITTI

<table>
<thead>
<tr>
<th>Method</th>
<th>Forward Pass (ms)</th>
<th>Total (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaserNet (Ours)</td>
<td>12</td>
<td>30</td>
</tr>
<tr>
<td>PIXOR [28]</td>
<td>35</td>
<td>62</td>
</tr>
<tr>
<td>PIXOR++ [27]</td>
<td>35</td>
<td>62</td>
</tr>
<tr>
<td>VoxelNet [30]</td>
<td>190</td>
<td>225</td>
</tr>
<tr>
<td>MV3D [51]</td>
<td>-</td>
<td>360</td>
</tr>
<tr>
<td>AVOD [15]</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>F-PointNet [22]</td>
<td>-</td>
<td>170</td>
</tr>
<tr>
<td>ContFuse [17]</td>
<td>60</td>
<td>-</td>
</tr>
</tbody>
</table>

LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving
Uber Advanced Technologies Group. CVPR 2019
Outline

• What is lidar?
• How do we make decisions about point clouds?
  • PointNet – orderless point processing
  • VoxelNet – voxel-based point processing
  • PointPillars – bird’s eye view point processing
    • Exploiting Visibility for 3D Object Detection
    • LaserNet – range image point processing
• PseudoLidar – Bird’s eye view depth map processing
Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving
Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger.
CVPR 2019
**Table 1:** 3D object detection results on the KITTI validation set. We report $\text{AP}_{\text{BEV}} / \text{AP}_{\text{3D}}$ (in %) of the car category, corresponding to average precision of the bird’s-eye view and 3D object box detection. Mono stands for monocular. Our methods with *pseudo-LiDAR* estimated by PSMNet* [3] (stereo) or DORN [10] (monocular) are in blue. Methods with LiDAR are in gray. Best viewed in color.

<table>
<thead>
<tr>
<th>Detection algorithm</th>
<th>Input signal</th>
<th>IoU = 0.5</th>
<th></th>
<th>IoU = 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy</td>
<td>Moderate</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>MONO3D [4]</td>
<td>Mono</td>
<td>30.5 / 25.2</td>
<td>22.4 / 18.2</td>
<td>19.2 / 15.5</td>
</tr>
<tr>
<td>MLF-MONO [33]</td>
<td>Mono</td>
<td>55.0 / 47.9</td>
<td>36.7 / 29.5</td>
<td>31.3 / 26.4</td>
</tr>
<tr>
<td>AVOD</td>
<td>Mono</td>
<td>61.2 / 57.0</td>
<td>45.4 / 42.8</td>
<td>38.3 / 36.3</td>
</tr>
<tr>
<td>F-POINTNET</td>
<td>Mono</td>
<td>70.8 / 66.3</td>
<td>49.4 / 42.3</td>
<td>42.7 / 38.5</td>
</tr>
<tr>
<td>3DOP [5]</td>
<td>Stereo</td>
<td>55.0 / 46.0</td>
<td>41.3 / 34.6</td>
<td>34.6 / 30.1</td>
</tr>
<tr>
<td>MLF-STEREO [33]</td>
<td>Stereo</td>
<td>-</td>
<td>53.7 / 47.4</td>
<td>-</td>
</tr>
<tr>
<td>AVOD</td>
<td>Stereo</td>
<td>89.0 / 88.5</td>
<td>77.5 / 76.4</td>
<td>68.7 / 61.2</td>
</tr>
<tr>
<td>F-POINTNET</td>
<td>Stereo</td>
<td>89.8 / 89.5</td>
<td>77.6 / 75.5</td>
<td>68.2 / 66.3</td>
</tr>
<tr>
<td>AVOD [17]</td>
<td>LiDAR + Mono</td>
<td>90.5 / 90.5</td>
<td>89.4 / 89.2</td>
<td>88.5 / 88.2</td>
</tr>
<tr>
<td>F-POINTNET [25]</td>
<td>LiDAR + Mono</td>
<td>96.2 / 96.1</td>
<td>89.7 / 89.3</td>
<td>86.8 / 86.2</td>
</tr>
</tbody>
</table>
Summary

• Popular CNN backbones aren’t a direct fit for 3D point processing tasks.
• It’s not clear how to use deep learning on 3D data
  • Use a truly permutation invariant representation (PointNet)
  • Use a voxel representation (VoxelNet)
  • Use a bird’s a view representation (PointPillars)
  • Create a range image (LaserNet)
• These alternate representations might be applicable more broadly, e.g. reasoning about depth estimates might be easier in bird’s eye view (PseudoLidar)