3D Point Processing and Lidar

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 couple of clever methods, typical of the literature as a whole, to work around this.
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
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

Stanford University
Big Data + Deep Representation Learning

Robot Perception

Augmented Reality

Shape Design

Emerging 3D Applications

source: Scott J Grunewald

source: Google Tango

source: solidsolutions
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)
3D Representation: Point Cloud

Point cloud is close to raw sensor data

LiDAR

Point Cloud

Depth Sensor
3D Representation: Point Cloud

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

LiDAR → Point Cloud → Mesh → Volumetric → Depth Map

Depth Sensor
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></td>
</tr>
<tr>
<td>FPFH</td>
<td>No</td>
<td>L</td>
<td>2.5D Scans (Pseudo single position range images)</td>
</tr>
<tr>
<td>VFH</td>
<td>No</td>
<td>G</td>
<td>Object detection with basic pose estimation</td>
</tr>
<tr>
<td>CVFH</td>
<td>No</td>
<td>R</td>
<td>Object detection with basic pose estimation, detection of partial objects</td>
</tr>
<tr>
<td>RIFT</td>
<td>Yes</td>
<td>L</td>
<td>Real world 3D-Scans with no mirror effects. RIFT is vulnerable against flipping.</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.

<table>
<thead>
<tr>
<th>Conversion</th>
<th>Deep Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voxelization</td>
<td>3D CNN</td>
</tr>
<tr>
<td>Projection/Rendering</td>
<td>2D CNN</td>
</tr>
<tr>
<td>Feature extraction</td>
<td>Fully Connected</td>
</tr>
</tbody>
</table>
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
...

PointNet
Our Work: PointNet

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

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:

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

... How can we construct a family of symmetric functions by neural networks?
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
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

\( h \)

\((1,2,3) \rightarrow \)

\((1,1,1) \rightarrow \)

\((2,3,2) \rightarrow \)

\(\vdots\)

\((2,3,4) \rightarrow \)
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:

\[ f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n)) \] is symmetric if \( g \) is symmetric.
Empirically, we use multi-layer perceptron (MLP) and max pooling:

\[
g(1,2,3) \xrightarrow{\text{MLP}} h \xrightarrow{\text{MLP}} \text{max} \xrightarrow{\text{MLP}} \gamma
\]

PointNet (vanilla)
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.
Input Alignment by Transformer Network

Idea: Data dependent transformation for automatic alignment

Diagram:
- **Data** (N) is transformed by the **T-Net** with transform params.
- The transformed data (N) is then transformed by the **Transform** module.
- The output is the **Transformed Data** (N).
Idea: Data dependent transformation for automatic alignment
Idea: Data dependent transformation for automatic alignment
The transformation is just matrix multiplication!
Embedding Space Alignment

T-Net

transform params: 64x64

Matrix Mult.

Input embeddings: Nx64

Transformed embeddings: Nx64
Embedding Space Alignment

**T-Net**

Transform parameters: 64x64

Matrix Mul.

Input embeddings: Nx64

Transformed embeddings: Nx64

Regularization:

Transform matrix A 64x64 close to orthogonal:

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

input points

nx3

input transform

nx3

mlp (64,64)

shared

nx64

feature transform

nx64

mlp (64,128,1024)

shared

nx1024

max pool

1024

global feature
PointNet Classification Network

input points $n \times 3$ → input transform $n \times 3$ → mlp (64,64) → feature transform $n \times 64$ → mlp (64,128,1024) → max pool $1024$ → global feature $512,256,k$ → output scores $k$
Extension to PointNet Segmentation Network

- input transform
- mlp (64,64)
- feature transform
- mlp (64,128,1024)
- max pool
- 1024
- mlp (512,256,k)

Local embedding
Global feature

T-Net
3x3 transform
matrix multiply

T-Net
64x64 transform
matrix multiply
Extension to PointNet Segmentation Network
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>3DShapeNets [29]</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>skateboard</th>
<th>table</th>
</tr>
</thead>
<tbody>
<tr>
<td># shapes</td>
<td></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>
</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>74.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>74.8</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

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

dataset: ModelNet40; metric: 40-class classification accuracy (%)

![Graph showing the relationship between accuracy and missing data ratio for different corruption methods.](image)
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?
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:
Which points won’t affect the global feature?
Visualizing Global Point Cloud Features

Original Shape:

Critical Point Set:

Upper bound set:
Visualizing Global Point Cloud Features (OOS)

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></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 (ModelNet shape classification)</td>
<td></td>
</tr>
<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

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
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

\[(x^g_c, y^g_c, z^g_c, l^g, w^g, h^g, \theta^g)\]
VoxelNet qualitative results
## VoxelNet quantitative results

<table>
<thead>
<tr>
<th>Method</th>
<th>Modality</th>
<th>Car</th>
<th>Pedestrian</th>
<th>Cyclist</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></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>
</tr>
<tr>
<td>3DOP [4]</td>
<td>Stereo</td>
<td>6.55</td>
<td>5.07</td>
<td>4.10</td>
</tr>
<tr>
<td>VeloFCN [22]</td>
<td>LiDAR</td>
<td>15.20</td>
<td>13.66</td>
<td>15.98</td>
</tr>
<tr>
<td>MV (BV+FV) [5]</td>
<td>LiDAR</td>
<td>71.19</td>
<td>56.60</td>
<td>55.30</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>
</tr>
<tr>
<td>HC-baseline</td>
<td>LiDAR</td>
<td>71.73</td>
<td>59.75</td>
<td>55.69</td>
</tr>
<tr>
<td>VoxelNet</td>
<td>LiDAR</td>
<td>81.97</td>
<td>65.46</td>
<td>62.85</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
Runtime / accuracy tradeoff
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

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

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

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

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

**Visibility Volume** → **Visibility-augmented Deep Voxel Representation**
Efficient Ray-casting via Voxel Traversal

Though animated in 2D, the idea generalizes in 3D.

A Fast Voxel Traversal Algorithm for Ray Tracing
John Amanatides, Andrew Woo
Eurographics 1987

3D Visibility Volume
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>Easy</th>
<th>Moderate</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaserNet (Ours)</td>
<td>LiDAR</td>
<td>78.25</td>
<td>73.77</td>
<td>66.47</td>
</tr>
<tr>
<td>PIXOR [28]</td>
<td>LiDAR</td>
<td>81.70</td>
<td>77.05</td>
<td>72.95</td>
</tr>
<tr>
<td>PIXOR++ [27]</td>
<td>LiDAR</td>
<td><strong>89.38</strong></td>
<td>83.70</td>
<td><strong>77.97</strong></td>
</tr>
<tr>
<td>VoxelNet [30]</td>
<td>LiDAR</td>
<td>89.35</td>
<td>79.26</td>
<td>77.39</td>
</tr>
<tr>
<td>MV3D [5]</td>
<td>LiDAR+RGB</td>
<td>86.02</td>
<td>76.90</td>
<td>68.49</td>
</tr>
<tr>
<td>AVOD [15]</td>
<td>LiDAR+RGB</td>
<td>88.53</td>
<td>83.79</td>
<td>77.90</td>
</tr>
<tr>
<td>F-PointNet [22]</td>
<td>LiDAR+RGB</td>
<td>88.70</td>
<td>84.00</td>
<td>75.33</td>
</tr>
<tr>
<td>ContFuse [17]</td>
<td>LiDAR+RGB</td>
<td>88.81</td>
<td><strong>85.83</strong></td>
<td>77.33</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><strong>12</strong></td>
<td><strong>30</strong></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 [5]</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 \text{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 PSMNEx* [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>( \text{IoU} = 0.5 )</th>
<th>( \text{IoU} = 0.7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy</td>
<td>Moderate</td>
<td>Hard</td>
</tr>
<tr>
<td><strong>MONO3D [4]</strong></td>
<td>Mono</td>
<td>30.5 / 25.2</td>
<td>22.4 / 18.2</td>
</tr>
<tr>
<td><strong>MLF-MONO [33]</strong></td>
<td>Mono</td>
<td>55.0 / 47.9</td>
<td>36.7 / 29.5</td>
</tr>
<tr>
<td><strong>AVOD</strong></td>
<td>Mono</td>
<td>61.2 / 57.0</td>
<td>45.4 / 42.8</td>
</tr>
<tr>
<td><strong>F-POINTNET</strong></td>
<td>Mono</td>
<td>70.8 / 66.3</td>
<td>49.4 / 42.3</td>
</tr>
<tr>
<td><strong>3DOP [5]</strong></td>
<td>Stereo</td>
<td>55.0 / 46.0</td>
<td>41.3 / 34.6</td>
</tr>
<tr>
<td><strong>MLF-STEREO [33]</strong></td>
<td>Stereo</td>
<td>-</td>
<td>53.7 / 47.4</td>
</tr>
<tr>
<td><strong>AVOD</strong></td>
<td>Stereo</td>
<td>89.0 / 88.5</td>
<td>77.5 / 76.4</td>
</tr>
<tr>
<td><strong>F-POINTNET</strong></td>
<td>Stereo</td>
<td>89.8 / 89.5</td>
<td>77.6 / 75.5</td>
</tr>
<tr>
<td><strong>AVOD [17]</strong></td>
<td>LiDAR + Mono</td>
<td>90.5 / 90.5</td>
<td>89.4 / 89.2</td>
</tr>
<tr>
<td><strong>F-POINTNET [25]</strong></td>
<td>LiDAR + Mono</td>
<td>96.2 / 96.1</td>
<td>89.7 / 89.3</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)