Lecture 9:
Deep Learning on Point Cloud for Shape Analysis
Instructor: Hao Su
Feb 6, 2018
Agenda

PointNet: A Basic Architecture for Point Cloud Processing

Using PointNet for 3D Object Detection
Image understanding: From feature engineering to learning

Feature engineering

SIFT
[Lowe, 1999]

Image gradients

Keypoint descriptor
Feature learning

Object classification accuracy on ImageNet (ILSVRC)

Feature engineering

Deep learning
Prior art: Handcrafted 3D features

Representatives:

- D2 [Osada, 2002]
- Spin Images [Johnson, 1999]

Cons:

- Hard Representation- Task-specific dependent
Fundamental challenge of 3D deep learning

Irregularity

Point cloud
(The most common 3D sensor data)

Mesh
(The most common modeling data)
Solution 1: Convert irregular to regular

High space/time complexity

Information loss in voxelization
Solution 2: Directly process point cloud data

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

![Diagram](image)
Properties of a desired neural network on point clouds

Point cloud: \( N \) **orderless** points, each represented by a \( D \) dim coordinate

2D array representation
Properties of a desired neural network on point clouds

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

2D array representation

Permutation invariance

Transformation invariance
Properties of a desired neural network on point clouds

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

2D array representation

Permutation invariance
Permutation invariance:

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

Examples:

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

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

\[ \ldots \]
Construct symmetric function family

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
Construct symmetric function family

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)

(1,1,1)

(2,3,2)

(2,3,4)
Construct symmetric function family

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

\[ h \]

\( (1,2,3) \)
\( (1,1,1) \)
\( (2,3,2) \)
\( (2,3,4) \)

\( g \)

simple 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.
Q: What symmetric functions can be constructed by PointNet?
A: Universal approximation to continuous symmetric functions

Theorem:

A Hausdorff continuous symmetric function \( f : 2^X \rightarrow \mathbb{R} \) can be arbitrarily approximated by PointNet.

\[
\left| f(S) - \gamma \left( \max_{x_i \in S} \{ h(x_i) \} \right) \right| < \epsilon
\]

\( S \subseteq \mathbb{R}^d \), \hspace{1cm} \text{PointNet (vanilla)}
Properties of a desired neural network on point clouds

Point cloud: $N$ orderless points, each represented by a $D$ dim coordinate.

Permutation invariance

Transformation invariance
Transformation invariance is desirable

Let $S$ be a shape. Then $f(T \cdot S) = f(S)$

$f$: classifier, $T$: transformation matrix
Incorporate transformer networks to input data

Input alignment to a canonical space
Point Feature Transform: Feature alignment to a canonical space

Feature alignment to a canonical space

Local point feature: \( \mathbf{N} \times K \)

Transform params: \( K \times K \)

Transformed feature: \( \mathbf{N} \times K \)
Efficiency of PointNet

Space complexity (#params)

Saves 95% GPU memory

MVCNN [Su et al. 2015]
Subvolume [Su et al. 2016]
VRN [Su et al. 2016]
PointNet [Su et al. 2017]
Efficiency of PointNet

Saves 95% time

Time complexity (FLOPs/sample)

- MVCNN: [Su et al. 2015]
- Subvolume: [Su et al. 2016]
- VRN: [Su et al. 2016]
- PointNet: [Su et al. 2017]
Efficiency of PointNet

A promising architecture for portable devices

Space complexity (#params)

- MVCNN
- Subvolume
- VRN
- PointNet

Time complexity (FLOPs/sample)

- MVCNN
- Subvolume
- VRN
- PointNet

Hao Su
Lecture 9 - 25
Robustness to data corruption

![Graph showing accuracy vs. missing data ratio for PointNet and VoxNet]
Robustness to data corruption

Segmentation from partial scans
Visualize what is learned by reconstruction

Salient points are discovered!
Agenda

PointNet: A Basic Architecture for Point Cloud Processing

Using PointNet for 3D Object Detection
Current State of Computer Vision

**2D Deep Learning**

Network Architectures:
- AlexNet, Network in Network, VGG, GoogleNet, STN, ResNet, DenseNet, ...

Frameworks for Recognition:
- R-CNN, Fast R-CNN, Faster-RCNN, SSD, YOLO, Feature Pyramid Network (FPN), Mask R-CNN etc.

**3D Deep Learning**

Network Architectures:
- VoxNet, Multi-view CNN, FPNN, Octree CNN, Kd-network, PointNet, PointNet++ etc.

?
Current State of Computer Vision

2D Deep Learning

Network Architectures:
AlexNet, Network in Network, VGG, GoogleNet, STN, ResNet, DenseNet, ...

Frameworks for Recognition:
R-CNN, Fast R-CNN, Faster-RCNN, SSD, YOLO, Feature Pyramid Network (FPN), Mask R-CNN etc.

3D Deep Learning

Network Architectures:
VoxNet, Multi-view CNN, FPNN, Octree CNN, Kd-network, PointNet, PointNet++ etc.

This work: A novel framework for 3D object detection with PointNet architectures.
What is 3D Object Detection?

**Input:** RGB-D data

“D” can be sparse point cloud from LiDAR or dense depth map from indoor depth sensors.

**Output:** Amodal 3D bounding boxes and semantic class labels for objects in the scene.

“amodal” means the 3D box is for the “complete” object even if part of it is invisible.
What is 3D Object Detection?
What is 3D Object Detection?

Figure from the recent VoxelNet paper from Apple.
What is 3D Object Detection?

*Figure from ICCV17 paper 2d-driven 3d object detection.*
Frustum PointNets for 3D Object Detection

- Leveraging mature 2D detectors for region proposal and 3D search space reduction
- Solving 3D detection problem with 3D data and 3D deep learning architectures
Our method ranks No. 1 on KITTI 3D Object Detection Benchmark

We get 5% higher AP than Apple’s recent CVPR submission and more than 10% higher AP than previous SOTA in easy category.
Our method ranks No. 1 on KITTI 3D Object Detection Benchmark

We are also 1st place for smaller objects (ped. and cyclist) winning with even bigger margins.

**Pedestrian**

<table>
<thead>
<tr>
<th>Method</th>
<th>Setting</th>
<th>Code</th>
<th>Moderate</th>
<th>Easy</th>
<th>Hard</th>
<th>Runtime</th>
<th>Environment</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-PointNet</td>
<td></td>
<td></td>
<td>44.89 %</td>
<td>51.21 %</td>
<td>40.23 %</td>
<td>0.17 s</td>
<td>GPU @ 3.0 Ghz (Python)</td>
<td></td>
</tr>
<tr>
<td>VxNet(LiDAR)</td>
<td></td>
<td></td>
<td>33.69 %</td>
<td>39.48 %</td>
<td>31.51 %</td>
<td>0.23 s</td>
<td>GPU @ 2.5 Ghz (Python + C/C++)</td>
<td></td>
</tr>
<tr>
<td>AVOD</td>
<td></td>
<td></td>
<td>25.87 %</td>
<td>32.67 %</td>
<td>25.01 %</td>
<td>0.08 s</td>
<td>Titan X (pascal)</td>
<td></td>
</tr>
<tr>
<td>3dSSD</td>
<td></td>
<td></td>
<td>17.35 %</td>
<td>20.22 %</td>
<td>17.20 %</td>
<td>0.03 s</td>
<td>GPU @ 2.5 Ghz (Python + C/C++)</td>
<td></td>
</tr>
</tbody>
</table>

**Cyclist**

<table>
<thead>
<tr>
<th>Method</th>
<th>Setting</th>
<th>Code</th>
<th>Moderate</th>
<th>Easy</th>
<th>Hard</th>
<th>Runtime</th>
<th>Environment</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-PointNet</td>
<td></td>
<td></td>
<td>56.77 %</td>
<td>71.96 %</td>
<td>50.39 %</td>
<td>0.17 s</td>
<td>GPU @ 3.0 Ghz (Python)</td>
<td></td>
</tr>
<tr>
<td>VxNet(LiDAR)</td>
<td></td>
<td></td>
<td>48.36 %</td>
<td>61.22 %</td>
<td>44.37 %</td>
<td>0.23 s</td>
<td>GPU @ 2.5 Ghz (Python + C/C++)</td>
<td></td>
</tr>
<tr>
<td>AVOD</td>
<td></td>
<td></td>
<td>30.43 %</td>
<td>43.74 %</td>
<td>30.12 %</td>
<td>0.08 s</td>
<td>Titan X (pascal)</td>
<td></td>
</tr>
</tbody>
</table>
Frustum-based 3D Object Detection

Challenges:
• Occlusions and clutters are common in frustum point cloud.
• Largely varying ranges of points in frustums.
Frustum PointNets

Frustum Proposal

3D Instance Segmentation

Amodal 3D Box Estimation

RGB image

Depth

2d region proposal

region2frustum

point cloud in frustum (n points)

3D Instance Segmentation PointNet

masking

segmented object points (m points)

T-Net

center residual translation

Amodal 3D Box Estimation PointNet

Box Parameters
It is the **3D field of view** of the notional camera.
Frustum Proposal

Input: RGB-D data
Frustum Proposal

Input: RGB-D data
Image region proposal

CNN

RGB image
Depth

2d region proposal
Frustum Proposal

Input: RGB-D data

Image region proposal

2D-3D lifting from depth map
Frustum Proposal

Input: RGB-D data

Image region proposal

2D-3D lifting from depth map

Frustum point cloud extraction
3D Instance Segmentation in Frustums

Localize object in frustum by point cloud segmentation.
3D Instance Segmentation in Frustums

Input: frustum point cloud
3D Instance Segmentation in Frustums

Input: frustum point cloud
Point cloud binary segmentation with PointNet: object of interest v.s. others
Amodal 3D Box Estimation

Estimate 3D bounding boxes from segmented object point clouds.
Amodal 3D Box Estimation

Input: object point cloud
Amodal 3D Box Estimation

Input: object point cloud
A regression PointNet estimates amodal 3D bounding box for the object
Frustum PointNets

In comparison with Mask R-CNN

**Mask R-CNN:** 2D box -> 2D segmentation

**Frustum PointNets:** 2D box -> 3D frustum -> 3D segmentation -> 3D amodal box
Frustum PointNets: Key to our Success

• **Representation.** We use PointNets for 3D estimation in raw point clouds.

• **Coordinates Normalization.** A series of coordinate transformations canonicalize the learning problems.

• **Loss function.** We design specialized loss functions for 3D bounding box regression.
Frustum PointNets: Key to our Success

• **Representation.** We use PointNets for 3D estimation in raw point clouds.

• **Coordinates Normalization.** A series of coordinate transformations canonicalize the learning problems.

• **Loss function.** We design specialized loss functions for 3D bounding box regression.
Representation Matters

Baseline by 2D Mask RCNN
Representation Matters

Baseline by 2D Mask RCNN

Ours
Frustum PointNets: Key to our Success

• **Representation.** We use PointNets for 3D estimation in raw point clouds.

• **Coordinates Normalization.** A series of coordinate transformations canonicalize the learning problems.

• **Loss function.** We design specialized loss functions for 3D bounding box regression.
Coordinates Normalization

(a) camera coordinate  
(b) frustum coordinate  
(c) 3D mask coordinate  
(d) 3D object coordinate
Coordinates Normalization

(a) camera coordinate  (b) frustum coordinate  (c) 3D mask coordinate  (d) 3D object coordinate
Coordinates Normalization

(a) camera coordinate
(b) frustum coordinate
(c) 3D mask coordinate
(d) 3D object coordinate

3D Instance Segmentation PointNet

Set Abstraction Layers
Point Feature Propagation Layers

frustum point cloud
(object of interest)

point cloud in frustum (n points)
segmented object points (m points)
T-Net
center/residual
translation
Amodal 3D Box Estimation PointNet
Box Parameters
Coordinates Normalization

(a) camera coordinate  
(b) frustum coordinate  
(c) 3D mask coordinate  
(d) 3D object coordinate

Frustum Proposal  
3D Instance Segmentation  
3D Instance Segmentation PointNet

RGB image  
Depth

2d region proposal  
region box  
point cloud in frustum (n points)  
segmented object points (m points)  
frustum rotation  
mask point centroid  
T-Net

Set Abstraction Layers  
Point Feature Propagation Layers

object point cloud (mask coordinate)  
center residual (mask coordinate)

3D Box Estimation PointNet

frustum point cloud (frustum coordinate)  
object of interest probability
Coordinates Normalization

Hao Su

Lecture 9 - 62
Coordinate Normalization

Table 7. **Effects of point cloud normalization.** Metric is 3D box estimation accuracy with IoU=0.7.
PointNet v2.0: Multi-Scale PointNet

N points in $(x, y)$

$N_1$ points in $(x, y, f)$

$N_2$ points in $(x, y, f')$

1. Larger receptive field in higher layers
2. Less points in higher layers (more scalable)
3. Weight sharing
4. Translation invariance (local coordinates in local regions)
Frustum PointNets: Key to our Success

- **Representation.** We use PointNets for 3D estimation in raw point clouds.

- **Coordinates Normalization.** A series of coordinate transformations canonicalize the learning problems.

- **Loss function.** We design specialized loss functions for 3D bounding box regression.
Qualitative Results
(on KITTI and SUN-RGBD)
Remarkable box estimation accuracy even with a dozen of points or with very partial point cloud
occluding traffic sign..
Inaccurate box regression with too few LiDAR points.

Image features could help.
Missing 2D detection results in no 3D detection

Multiple ways for proposal could help (e.g. bird’s eye view, multiple 2D proposal networks)
Strong occlusion. Just 4 LiDAR points..

Challenging case for instance segmentation (multiple closeby objects in a single frustum)
Missed 2D detection in a complicated scene with strong occlusions

Challenging segmentation case
### Table 5. 3D object detection AP on SUN-RGBD val set.

<table>
<thead>
<tr>
<th></th>
<th>bathtub</th>
<th>bed</th>
<th>bookshelf</th>
<th>chair</th>
<th>desk</th>
<th>dresser</th>
<th>nightstand</th>
<th>sofa</th>
<th>table</th>
<th>toilet</th>
<th>Runtime</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSS [35]</td>
<td>44.2</td>
<td>78.8</td>
<td>11.9</td>
<td>61.2</td>
<td>20.5</td>
<td>6.4</td>
<td>15.4</td>
<td>53.5</td>
<td>50.3</td>
<td>78.9</td>
<td>19.55s</td>
<td>42.1</td>
</tr>
<tr>
<td>COG [30]</td>
<td>58.3</td>
<td>63.7</td>
<td>31.8</td>
<td>62.2</td>
<td><strong>45.2</strong></td>
<td>15.5</td>
<td>27.4</td>
<td>51.0</td>
<td><strong>51.3</strong></td>
<td>70.1</td>
<td><strong>10-30min</strong></td>
<td>47.6</td>
</tr>
<tr>
<td>2D-driven [16]</td>
<td>43.5</td>
<td>64.5</td>
<td>31.4</td>
<td>48.3</td>
<td>27.9</td>
<td>25.9</td>
<td>41.9</td>
<td>50.4</td>
<td>37.0</td>
<td>80.4</td>
<td>4.15s</td>
<td>45.1</td>
</tr>
<tr>
<td>Ours (v1)</td>
<td>43.3</td>
<td><strong>81.1</strong></td>
<td><strong>33.3</strong></td>
<td><strong>64.2</strong></td>
<td>24.7</td>
<td><strong>32.0</strong></td>
<td><strong>58.1</strong></td>
<td><strong>61.1</strong></td>
<td>51.1</td>
<td><strong>90.9</strong></td>
<td><strong>0.12s</strong></td>
<td><strong>54.0</strong></td>
</tr>
</tbody>
</table>

Evaluation metric is average precision with 3D IoU threshold 0.25 as proposed by [33]. Note that both COG [30] and 2D-driven [16] use room layout context to boost performance while ours and DSS [35] not. Compared with previous state-of-the-arts our method is 6.4% to 11.9% better in mAP as well as one to three orders of magnitude faster.
Opening in my Lab for Shape Processing

• Task: to make ShapeNet amiable for machine learning researchers (ShapeNet v2.0)
• You will gain a lot of experience for geometry processing
• Not much research into machine learning in the beginning, though, but
  • Can attend my group meetings
  • May have the opportunity to work on learning stuff in the future
  • Acknowledged as in the ShapeNet team
• Requirement:
  • Very strong programming ability
  • Past CG experience
  • Master thesis topic