CS468: 3D Deep Learning on Point Cloud Data

Hao Su
Stanford University

May 10, 2017
3D perception from a single image
Monocular vision

A typical prey

A typical predator

Cited from https://en.wikipedia.org/wiki/Binocular_vision
A psychological evidence – mental rotation

by Roger N. Shepard, National Science Medal Laurate and Lynn Cooper, Professor at Columbia University
Visual cues are complicated

- contrast
- color
- texture
- motion
- symmetry
- part
- category-specific 3D knowledge

......
Status review of monocular vision algorithms

- Shape from X (texture, shading, …)

[Horn, 1989]

[Kender, 1979]
Status review of monocular vision algorithms

- Shape from X (texture, shading, …)

  [Horn, 1989]

- Learning-based (from small data)

  [Kender, 1979]

  Hornem et al, ICCV’05
  Saxena et al, NIPS’05

  - large planes
  - fine structure
  - topological variation
  - …
Status review of monocular vision algorithms

- Shape from X (texture, shading, …)
- Learning-based (from small data)

### Shape from X

- [Horn, 1989]
- [Kender, 1979]

### Learning-based

- Hoiem et al, ICCV'05
- Saxena et al, NIPS'05
- ... (includes examples)

**Strong assumption**

**Not robust**

- large planes
- fine structure
- topological variation
- ...

...
Data-driven 2D-3D lifting

Many 3D objects

A priori knowledge of the 3D world
Two pillars for learning-based 3D understanding
Outline

- Motivation
- Data
  - Algorithms for 2D-3D lifting
  - Algorithm for 3D representation learning
Need many 2D images with 3D labels
Accurate 3D label annotation is expensive

Example: viewpoint estimation

PASCAL3D+ dataset [Xiang et al.]
Accurate 3D label annotation is expensive

Step 1:
Choose similar model
Accurate 3D label annotation is expensive

**Step1:**
Choose similar model

**Step2:**
Coarse Viewpoint Labeling
Accurate 3D label annotation is expensive

Step 1:
Choose similar model

Step 2:
Coarse Viewpoint Labeling

Step 3:
Label keypoints For alignment

Annotation takes ~1 min per object
High-cost Label Acquisition vs. High-capacity Model

Mismatch!

Needs millions of images to train

How to get MORE images with ACCURATE labels?
Manual labeling by annotators

Auto labeling through rendering
Good News: ShapeNet

http://shapenet.cs.stanford.edu

millions of shapes with geometric and physical annotations 
(ongoing efforts)
Synthesize for learning

ShapeNet

Rendering

Training

Shape (3D form)

Learner

Synthetic Images (2D form)

[Su et al., RenderForCNN, ICCV15]
Synthesize for learning

ShapeNet → Rendering → Synthetic Images (2D form) → Learner → Training → Shape (3D form) → Deep Neural Network

[Su et al., RenderForCNN, ICCV15]
How shall we represent 3D shapes?

- **ShapeNet**
- **Rendering**
- **Learner**
- **Training**
  - Shape (3D form)
  - Synthetic Images (2D form)

[Su et al., RenderForCNN, ICCV15]
Many 3D representations are available

Candidates:

multi-view images
depth map
volumetric
polygonal mesh
point cloud
primitive-based CAD models
3D representation

Candidates:
- multi-view images
- depth map
- volumetric
- polygonal mesh
- point cloud
- primitive-based CAD models

Novel view image synthesis

[Su et al., ICCV15]
[Dosovitskiy et al., ECCV16]
3D representation

Candidates:

- multi-view images
- depth map
- volumetric
- polygonal mesh
- point cloud
- primitive-based CAD models
3D representation

Candidates:

- multi-view images
- depth map
- **volumetric**
- polygonal mesh
- point cloud
- primitive-based CAD models
3D representation

Candidates:

- multi-view images
- depth map
- volumetric
- **polygonal mesh**
- point cloud
- primitive-based CAD models
3D representation

Candidates:

- multi-view images
- depth map
- volumetric
- polygonal mesh
- point cloud
- primitive-based CAD models
3D representation

a chair assembled by cuboids

Candidates:

- multi-view images
- depth map
- volumetric
- polygonal mesh
- point cloud
- primitive-based CAD models
Two groups of representations

**Rasterized form**
(regular grids)

- Multi-view images
- Depth map
- Volumetric

**Geometric form**
(irregular)

- Polygonal mesh
- Point cloud
- Primitive-based CAD models
Extant 3D DNNs work on grid-like representations.

Candidates:
- multi-view images
- depth map
- volumetric
- polygonal mesh
- point cloud
- primitive-based CAD models
It is possible to aggregate information from multiple views
Recurrent Neural Network

[Christopher Olah] Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
[Christopher Olah] Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
It is possible to aggregate information from multiple views.
Training

• Voxel-wise cross entropy loss

\[ L(\mathcal{X}, y) = \sum_{i,j,k} y_{(i,j,k)} \log(p_{(i,j,k)}) + (1 - y_{(i,j,k)}) \log(1 - p_{(i,j,k)}) \]

• ShapeNet
  • 50k CAD models
  • Render from arbitrary views
  • Random number of images w/ random order
  • Random background, translation
Ideally, a 3D representation should be

Friendly to learning

• easily formulated as the output of a neural network
• fast forward-/backward- propagation
• etc.
Ideally, a 3D representation should be

**Friendly to learning**
- easily formulated as the output of a neural network
- fast forward-/backward- propagation
- etc.

**Flexible**
- can precisely model a great variety of shapes
- etc.
Ideally, a 3D representation should be

**Friendly to learning**
- easily formulated as the output of a neural network
- fast forward-/backward- propagation
- etc.

**Flexible**
- can precisely model a great variety of shapes
- etc.

**Geometrically manipulable**
- geometrically deformable, interpolable and extrapolable for networks
- convenient to impose structural constraints
- etc.

**Others**
### The problem of grid representations

<table>
<thead>
<tr>
<th></th>
<th>Affability to learning</th>
<th>Flexibility</th>
<th>Geometric manipulability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multi-view images</strong></td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **Volumetric occupancy**  | ✗                      | ✓           | ✗                        | Expensive to compute: $O(N^3)$
|                           |                        |             |                          |
| **Depth map**             | ✓                      | ✗           | ✓                        | Cannot model “back side”
**Typical artifacts of volumetric reconstruction**

<table>
<thead>
<tr>
<th>Multi-view images</th>
<th>Affability</th>
<th>Geometric manipulability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volumetric occupancy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Depth map</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Missing thin structures due to **improper shape space structure**
- Hard for the network to rotate / deform / interpolate
How about learning to predict geometric forms?

Rasterized form (regular grids)

Geometric form (irregular)

Candidates:

- multi-view images
- depth map
- volumetric
- polygonal mesh
- point cloud
- primitive-based CAD models
Outline

- Motivation
- Data
- Algorithms for 2D-3D lifting
- Shape abstraction by volumetric primitives
Our result: 3D reconstruction from real Images

CVPR 2017, A Point Set Generation Network for 3D Object Reconstruction from a Single Image

Input

Reconstructed 3D point cloud
Our result: 3D reconstruction from real Images

CVPR 2017, A Point Set Generation Network for 3D Object Reconstruction from a Single Image
Flexible

• a few thousands of points can precisely model a great variety of shapes
3D point clouds

- Flexible
  - a few thousands of points can precisely model a great variety of shapes

- Geometrically manipulable
  - deformable
  - interpolable, extrapolable
  - convenient to impose structural constraints
Pipeline

2K object categories
200K shapes
~10M image/point set pairs
Pipeline

Shape predictor

Prediction

\( \{ (x'_1, y'_1, z'_1) \}
\( (x'_2, y'_2, z'_2) \)
\( \ldots \)
\( (x'_n, y'_n, z'_n) \) \}

Groundtruth point set

CVPR '17, Point Set Generation
Pipeline

Shape predictor

\[ (f) \]

Prediction

\[
\begin{align*}
(x'_1, y'_1, z'_1) \\
(x'_2, y'_2, z'_2) \\
\vdots \\
(x'_n, y'_n, z'_n)
\end{align*}
\]

A set is invariant up to permutation

Groundtruth point set

CVPR '17, Point Set Generation
Pipeline

Shape predictor $(f)$

Prediction

Groundtruth point set

Loss on sets $(L)$

CVPR '17, Point Set Generation
Pipeline

Shape predictor $(f)$

Prediction

\[
\{(x'_1, y'_1, z'_1), (x'_2, y'_2, z'_2), \ldots, (x'_n, y'_n, z'_n)\}
\]

Loss on sets $(L)$

Groundtruth point set

CVPR '17, Point Set Generation
Set comparison

Given two sets of points, measure their discrepancy
Set comparison

Given two sets of points, measure their discrepancy

Key challenge:
correspondence problem
Correspondence (I): optimal assignment

Given two sets of points, measure their discrepancy

a.k.a Earth Mover’s distance (EMD)

\[ d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \rightarrow S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2 \text{ where } \phi : S_1 \rightarrow S_2 \text{ is a bijection.} \]
Correspondence (II): closest point

Given two sets of points, measure their discrepancy

\[ d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|^2_2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|^2_2 \]

a.k.a Chamfer distance (CD)
Required properties of distance metrics

Geometric requirement

Computational requirement
Required properties of distance metrics

Geometric requirement

- Reflects natural shape differences
- Induce a nice space for *shape interpolations*

Computational requirement
How distance metric affects learning?

A fundamental issue: inherent ambiguity in 2D-3D dimension lifting
How distance metric affects learning?

A fundamental issue: inherent ambiguity in 2D-3D dimension lifting
How distance metric affects learning?

A fundamental issue: inherent ambiguity in 2D-3D dimension lifting
A fundamental issue: inherent ambiguity in 2D-3D dimension lifting

- By loss minimization, the network tends to predict a “mean shape” that averages out uncertainty
Distance metrics affect mean shapes

The mean shape carries characteristics of the distance metric

$$\bar{x} = \arg\min_x E_{s \sim S}[d(x, s)]$$

continuous hidden variable (radius)

Input  EMD mean  Chamfer mean

CVPR '17, Point Set Generation
Mean shapes from distance metrics

The mean shape carries characteristics of the distance metric

\[ \bar{x} = \arg \min_x E_{s \sim S}[d(x, s)] \]

- **Input**
- **EMD mean**
- **Chamfer mean**

**Continuous hidden variable** (radius)

**Discrete hidden variable** (add-on location)

*CVPR '17, Point Set Generation*
Comparison of predictions by EMD versus CD

Input

EMD

Chamfer

CVPR '17, Point Set Generation
Required properties of distance metrics

Geometric requirement

• Reflects natural shape differences
• Induce a nice space for shape interpolations

Computational requirement

• Defines a loss function that is numerically easy to optimize
Computational requirement of metrics

To be used as a loss function, the metric has to be

• **Differentiable** with respect to point locations

• **Efficient** to compute
Computational requirement of metrics

- **Differentiable** with respect to point location

Chamfer distance

\[ d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \]

Earth Mover’s distance

\[ d_{EMD}(S_1, S_2) = \min_{\phi : S_1 \rightarrow S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2 \quad \text{where } \phi : S_1 \rightarrow S_2 \text{ is a bijection.} \]

- Simple function of coordinates
- In general positions, the correspondence is unique
- With infinitesimal movement, the correspondence does not change

**Conclusion:** differentiable almost everywhere
Computational requirement of metrics

- **Differentiable** with respect to point location

  - For many *algorithms* (sorting, shortest path, network flow, ...),
  - an infinitesimal change to model parameters (almost) does not change solution structure,

  leads to differentiable a.e.!
Computational requirement of metrics

- **Efficient** to compute

  Chamfer distance: trivially parallelizable on CUDA

  Earth Mover’s distance (optimal assignment):
    - We implement a *distributed* approximation algorithm on CUDA
    - Based upon [Bertsekas, 1985], \((1 + \epsilon)\)-approximation

*Bertsekas, 1985*
Pipeline

Deep network 

\((f)\)

Prediction

\[ \{(x_1, y_1, z_1) \}
\[ \{(x_2, y_2, z_2) \}
\[ \ldots \}
\[ \{(x_n, y_n, z_n) \}
\]

Loss on sets 

\((L)\)

CVPR '17, Point Set Generation
Deep neural network

Universal function approximator

- A cascade of layers
Deep neural network

Universal function approximator

- A cascade of layers
- Each layer conducts a simple transformation (parameterized)
Deep neural network

- A cascade of layers
- Each layer conducts a simple transformation (parameterized)
- Millions of parameters, has to be fitted by many data
Pipeline

Deep network $(f)$

Loss on sets $(L)$

CVPR '17, Point Set Generation
Pipeline

Encoder → shape embedding → Predictor

\{ (x_1, y_1, z_1) \\
(x_2, y_2, z_2) \\
\ldots \\
(x_n, y_n, z_n) \}

Loss on sets \((L)\)

CVPR '17, Point Set Generation
Pipeline

Encoder → shape embedding → Predictor → \{ (x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n) \} → Loss on sets \((L)\)

CVPR '17, Point Set Generation
Pipeline

Loss on sets $(L)$

CVPR '17, Point Set Generation
Pipeline

Conv

Predictor

\{ (x_1, y_1, z_1) \\ (x_2, y_2, z_2) \\ \ldots \\ (x_n, y_n, z_n) \}

Loss on sets $(L)$

sample
Natural statistics of geometry

- Many local structures are common
  - e.g., planar patches, cylindrical patches
  - strong local correlation among point coordinates
• Many local structures are common
  • e.g., planar patches, cylindrical patches
  • **strong local correlation** among point coordinates
• Also some intricate structures
  • points have **high local variation**
Pipeline

Capture common structures

Encode (Conv)

Deconv branch

\[ \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\} \]

FC branch

\[ \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\} \]

Capture intricate structures

Loss on sets \((L)\)
Pipeline

Capture common structures

Capture intricate structures

Deconv branch

FC branch

Loss on sets \( (L) \)

\[
\begin{align*}
\{ (x_1, y_1, z_1) \\ (x_2, y_2, z_2) \\ \ldots \\ (x_n, y_n, z_n) \}
\end{align*}
\]

\[
\begin{align*}
\{ (x_1, y_1, z_1) \\ (x_2, y_2, z_2) \\ \ldots \\ (x_n, y_n, z_n) \}
\end{align*}
\]
Review: deconv network

- Output $n$D arrays, e.g., 2D segmentation map
- **Common local patterns** are learned from data
- Predict **locally correlated** data well
- Weight sharing reduces the number of params

Credit: FCNN, Long et al.
Review: deconv network

- Output nD arrays, e.g., 2D segmentation map
- **Common local patterns** are learned from data
- Predict **locally correlated** data well
- Weight sharing reduces the number of params

**How to predict curved surfaces in 3D?**

Deconv network for image segmentation

*Credit: FCNN, Long et al.*
• Surface parametrization (2D $\leftrightarrow$ 3D mapping)

Credit: Discrete Differential Geometry, Crane et al.
Prediction of curved 2D surfaces in 3D

- Surface parametrization (2D ↔ 3D mapping)

Credit: Discrete Differential Geometry, Crane et al.
Prediction of curved 2D surfaces in 3D

- Surface parametrization (2D ↔ 3D mapping)

Credit: Discrete Differential Geometry, Crane et al.
Parametrization prediction by deconv network

Capture common structures

Capture intricate structures

Deconv branch

FC branch

\[
\{(x_1, y_1, z_1) \\
(x_2, y_2, z_2) \\
\ldots \\
(x_n, y_n, z_n)\}
\]

\[
\{(x_1, y_1, z_1) \\
(x_2, y_2, z_2) \\
\ldots \\
(x_n, y_n, z_n)\}
\]

\[
\{(x_1, y_1, z_1) \\
(x_2, y_2, z_2) \\
\ldots \\
(x_n, y_n, z_n)\}
\]

Loss on sets \(L\)
Parametrization prediction by deconv network

Capture common structures

Capture intricate structures

conv

deconv

FC branch

coordinate maps

\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\}

\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\}

\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\}

Loss on sets \((L)\)

CVPR '17, Point Set Generation
Parametrization prediction by deconv network

Capture common structures

deconv

conv

Note that
- The parametrization (2D/3D mapping) is learned from data
- i.e., obtains a network and data friendly parametrization

CVPR '17, Point Set Generation

coordinate maps

\{ (x_1, y_1, z_1) \\
(x_2, y_2, z_2) \\
\ldots \\
(x_n, y_n, z_n) \}

\{ (x_1, y_1, z_1) \\
(x_2, y_2, z_2) \\
\ldots \\
(x_n, y_n, z_n) \}

\( L \)

Loss on sets
Visualization of the learned parameterization

Observation:

- Learns a **smooth** parametrization
- Because deconv net tends to predict data with local correlation

(map of x coord) (map of y coord) (map of z coord)
Visualization of the learned parameterization

Observation:
• Learns a **smooth** parametrization
• Because deconv net tends to predict data with local correlation
• Corresponds to **smooth surfaces**!

(map of x coord) -> (x_k, y_k, z_k) -> (map of y coord) -> (map of z coord) -> 3D model
Natural statistics of geometry

- Many local structures are common
  - e.g., planar patches, cylindrical patches
  - **strong local correlation** among point coordinates
- Also some intricate structures
  - points have **high local variation**
Pipeline

Capture common structures

Capture intricate structures

Loss on sets ($L$)
Pipeline

Capture common structures

Capture intricate structures

Conv

Deconv

Fc

Loss on sets

(L)

CVPR '17, Point Set Generation
Points are predicted **independently**

Dense connection introduces **more parameters** to accommodate **high variance**
Visualization of the effect of FC branch

Observation:
• The arrangement of predicted points are uncorrelated
Visualization of the effect of FC branch

Observation:
- The arrangement of predicted points are uncorrelated
- Located at **fine** structures

x-coord  y-coord  z-coord  red
Q: Which color corresponds to the deconv branch? FC branch?
Q: Which color corresponds to the deconv branch? FC branch?

**blue**: deconv branch – *large, smooth* structures

**red**: FC branch – *intricate* structures
Effect of combining two branches

Train/tested on 2K object categories

- Chamfer Distance (Error)
  - Deconv only: 0.525
  - FC only: 0.45
  - Combined: 0.3

CVPR '17, Point Set Generation
Real-world results

CVPR '17, Point Set Generation
Generalization to unseen categories

Out of training categories

CVPR '17, Point Set Generation
Comparison to state-of-the-art

- Better global structure
- Better details
Comparison to state-of-the-art

Trained/tested on 2K object categories

Chamfer Distance (Error)

Baseline (mean shape) [Choy et. al, ECCV16] Ours

CVPR ’17, Point Set Generation
Extension: shape completion for RGBD data

RGBD map (input)  90° view of input  output: completed point cloud

CVPR '17, Point Set Generation
Open problems

A better metric that takes the best of Chamfer and EMD?

How to add further structure constraints?

How to extend the pipeline to scene level?

How generalizable the method is?

In principle, what is the generalizability of a geometry estimator? To what extend is 3D perception ability innate or learned?
Outline

- Motivation
- Data
- Algorithms for 2D-3D lifting
- Shape abstraction by volumetric primitives
How about learning to predict geometric forms?

Rasterized form (regular grids)

Geometric form (irregular)

Candidates:
- multi-view images
- depth map
- volumetric
- polygonal mesh
- point cloud
- primitive-based CAD models
We learn to predict a corresponding shape composed by primitives. It allows us to predict consistent compositions across objects.
Unsupervised parsing

Each point is colored according to the assigned primitive
A historical overview

*Generalized Cylinders*, Binford (1971)
We predict primitive parameters: size, rotation, translation of M cuboids.
We predict primitive parameters: size, rotation, translation of M cuboids.

Variable number of parts? We predict “primitive existence probability”
Loss function
Loss function construction

Basic idea: **Chamfer distance**!

\[
d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2
\]
Loss function construction

Sample points on the groundtruth mesh and predicted assembly

Each point is a **linear function** of mesh/primitive vertex coordinates

Differentiable!
Loss function construction

Sample points on the groundtruth mesh and predicted assembly

\[ \Delta(x, y) + \Delta(x', y') + \Delta(x'', y'') + \ldots + \Delta(x'', y'') \]

Each point is a **linear function** of mesh/primitive vertex coordinates

**Differentiable!**

Speed up the computation leveraging parameterization of primitives
Consistent primitive configurations

Primitive locations are consistent due to the smoothness of primitive prediction network
Unsupervised parsing

<table>
<thead>
<tr>
<th>Method</th>
<th>[31] (initial)</th>
<th>[31] (refined)</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>78.6</td>
<td>84.8</td>
<td>89.0</td>
</tr>
</tbody>
</table>

Mean accuracy (face area) on Shape COSEG chairs.
Shapes become more parsimonious as training progresses (due to our parsimony reward)
Image-based modeling
Open problems

How to introduce other primitives types?

Towards image based modeling, how to add more operations to edit those primitives?
  • e.g., Deform? Extrude? Loop cut?

How to use it for design purposes? For example, with certain structural and functional constraints.

Ultimately, we expect to automate the modeling process from images, as artists do.
Resources

• For both works, paper and source codes are available.


• Shubham Tulsiani, Hao Su, Leonidas Guibas, Alexei Efros, Jitendra Malik, Learning Shape Abstractions by Assembling Volumetric Primitives, CVPR2017 (oral)
To sum up

We explore to generate geometric representations by neural networks.

Point cloud based reconstruction has better quality than state-of-the-art volumetric shape generators.

Primitive-based CAD models can be generated to abstract polygonal meshes in an unsupervised manner.

Keys:
- network structure leveraging geometric natural statistics
- loss function design