Deep Learning for 3D
Toward Surface Generation

Thibault GROUEIX, Pierre-Alain LANGLOIS
Why learn?

1. Get rid of hand crafted priors - Manhattan world assumption [Furukawa2009]

2. Discover complex prior from data itself - Discovering 3D from sketch [Delanoy2017]
Data types

What kind of data/sensor is relevant as input for 3d reconstruction?

RGB Image(s)

RGBD Image(s)

PointCloud
Typical learning framework based on synthetic data

Partial Data $\mathbf{X}$ → Shape predictor → Loss $L$ → 3D Object $\mathbf{Y}$
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Shape predictor

Loss $L$
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Encoder $E$

Decoder $D$

Feature vector

Loss

$L(D(E(X)), Y)$
Training setup for 3D reconstruction

Partial Data \( X \)
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object \( Y \)

Choice of representation ?

\[ L(D(E(X)), Y) \]

Loss
Representations

Obvious in 2D...

Not so obvious in 3D!
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

3D Object $Y$

$E$ (Encoder)
- Feature vector

$D$ (Decoder)

$L(D(E(X)), Y)$ (Loss)

$X$: Partial Data
$Y$: 3D Object

$E$: Encoder
$D$: Decoder

Feature vector

$L$: Loss function

Thibault Groueix, Pierre-Alain Langlois, 2019
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

Loss

$L(D(E(X)), Y)$
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

3D Object $Y$

$L(D(E(X)), Y)$
Encoders for RGB & RGBD images

Do not reinvent the wheel:
Use state-of-the-art 2D networks
Do not reinvent the wheel:
Use state-of-the-art 2D networks
- Resnet [He2015] -> Skip connections
- BatchNorm [Ioffe2015]
Resnet 34 [He2015]
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Loss
$L(D(E(X)), Y)$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

Encoder $E$

Feature vector

Decoder $D$
PointNet [Qi2017]
Input pointcloud $\mathbf{X} = (x_1, x_2, \ldots, x_n)$
Input pointcloud $X = (x_1, x_2, \ldots, x_n)$

$E((x_1, x_2, \ldots, x_n)) = x_1, \ldots, x_n$

PointNet [Qi2017]
\[ E((x_1, x_2, \ldots, x_n)) = h(x_1), \ldots, h(x_n) \]
\[ E((x_1, x_2, \ldots, x_n)) = g(h(x_1), \ldots, h(x_n)) \]
\[ E((x_1, x_2, \ldots, x_n)) = \gamma(g(h(x_1), \ldots, h(x_n))) \]
\[ E((x_1, x_2, \ldots, x_n)) = \gamma( g( h(x_1), \ldots, h(x_n) ) ) \]
Results: Unified framework for various tasks

Credit [Qi2017]
PointNet Limitations  

Credit [Qi2017]

- Hierarchical Feature Learning
- Increasing receptive field

Global Feature Learning  
Receptive field: one point OR all points

V.S.

3D CNN (Wu et al.)

PointNet (vanilla) (Qi et al.)
Key idea: Global information is computed in 1 stage: the max function.

PointNet Module
Key idea: Global information is computed in 1 stage: the max function.

Encoder $E$ = PointNet Module
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?

Key considerations:
❖ Define a receptive field: Ball Query (PointNet++ \cite{qi2017pointnet++, simonovsky2017dynamic})? Nearest Neighbors? Nearest Neighbors in 8 quadrant (pointSIFT \cite{jiang2018point})?
❖ Choose a metric: Euclidean? Geodesic?
❖ Choose the features: 3D input space features? Current Layer features (Dynamic Graph CNN \cite{wang2018dynamic})?
❖ Global coordinates? Local coordinates \cite{qi2017pointnet++, wang2018dynamic}?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?

Key considerations:
- Define a receptive field: Ball Query (PointNet++ [Qi2017b, Simonovsky2017]) ? Nearest Neighbors? Nearest Neighbors in 8 quadrant (pointSIFT [Jiang2018])?
- Choose a metric: Euclidean? Geodesic?
- Choose the features: 3D input space features? Current Layer features (Dynamic Graph CNN [Wang2018])?
- Global coordinates? Local coordinates [Qi2017b, Wang2018]?
Key idea: Global information is computed in 1 stage: the max function. Inspired by their success in images, can we build hierarchical filters?

Key considerations:
❖ Define a receptive field: Ball Query (PointNet++ [Qi2017b, Simonovsky2017])? Nearest Neighbors? Nearest Neighbors in 8 quadrant (pointSIFT [Jiang2018])?
❖ Choose a metric: Euclidean? Geodesic?
❖ Choose the features: 3D input space features? Current Layer features (Dynamic Graph CNN [Wang2018])?
❖ Global coordinates? Local coordinates [Qi2017b, Wang2018]?
Key idea: Global information is computed in 1 stage: the max function.
Inspired by their success in images, can we build hierarchical filters?

Key considerations:
❖ Define a receptive field: Ball Query (PointNet++ [Qi2017b, Simonovsky2017]) ? Nearest Neighbors ? Nearest Neighbors in 8 quadrant (pointSIFT [Jiang2018]) ?
❖ Choose a metric: Euclidean ? Geodesic ?
❖ Choose the features: 3D input space features ? Current Layer features (Dynamic Graph CNN [Wang2018]) ?
❖ Global coordinates ? Local coordinates [Qi2017b, Wang2018]?
A number of (good) alternatives exists

- KD-Trees: [Klokov2017]
- PCPNet [Guerrero2017]
- Large-scale PointClouds: SuperPointGraph [Landrieu2018]
- Build a graph on your pointcloud and apply Graph Neural Networks: SyncSpecNet [Yi2016]
- Projection on enclosing sphere and equivariant convolutions from SO(3) [Esteves2018, Cohen2018]
Training setup for 3D reconstruction

**Partial Data** \( X \)
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

**3D Object** \( Y \)

**Encoder** \( E \)
- Feature vector

**Decoder** \( D \)

**Choice of representation**
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

**Loss**

\[ L(D(E(X)), Y) \]
Voxels 3d-r2n2 [Choy2016], Voxnet [Maturana2015], [Qi2016], [Wu2015]

- A 3D regular grid which subdivides a bounding box in the 3D space
- Allows direct generalization of the 2D methods (convolutions, pooling)
- Subject to the curse of dimensionality: memory inefficient
Volumetric representations

Encoder

100x100x100

50x50x50

20x20x20

Decoder
Hybrid Grid-Octree Data Structure

Octnet [Riegler2017], OGN [Tatarchenko2017]

➔ Grid of octrees with fixed small depth: typically 3

➔ Computationally more effective

➔ Good compression rate
If a cell contains data from the mesh, it takes value 1 and it is subdivided.

Otherwise, it takes the value 0.

Easy to compare with the L2 distance over voxels.

Figure 8: Voxelized 3D Shapes from ModelNet10.
Convolutions on Grid-Octree Data Structure

→ Improvement: Inside a given cell the convolution result is the same. We can compute it once.

→ Convolution is computed on the boundaries

---

Figure 14: Efficient Convolution.
Voxels at maximum resolution are pulled.

Voxels at higher resolutions are halved in size.

Encoder $E$

2D example

3D

(a) Input

(b) Output
Training setup for 3D reconstruction

Partial Data $\mathbf{X}$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $\mathbf{Y}$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

$L(\mathbf{D}(\mathbf{E}(\mathbf{X})), \mathbf{Y})$
Objective: Predicting the occupancy value of each cell in the octree

Issue: Contrarily to voxels, the octree structure is specific to each sample

→ We need to predict the octree structure
Unpooling on Grid-Octree Data Structure

→ All nodes double their sizes
Unpooling on Grid-Octree Data Structure

➔ All nodes double their sizes

What about capturing details at finer resolution?
Unpooling on Grid-Octree Data Structure

➔ All nodes double their sizes

What about capturing details at finer resolution?

➔ If autoencoder, we can subdivide according to the input octree’s structure.

➔ In the case of single image reconstruction, there is a need to know whether terminal voxels can be split into 8 to capture finer details [Tatarchenko2017]
Octree generating networks - results

Subdivision is predicted as a classification task. [Tatarchenko2017]

The red cell can either be
- full or empty: we don’t subdivide
- mixed: we subdivide

This can be supervised at each layer of the network because we know whether a subdivision occurs or not in the ground truth.
Octree generating networks - results

OGN [Tatarchenko2017]

Input | $32^3$ | $64^3$ | $128^3$ | $256^3$ | GT $256^3$

Figure 8. Single-image 3D reconstruction on the ShapeNet-cars dataset using OGN in different resolutions.
Octree-based reconstruction

→ Gives insights regarding the extension of network operations to 3D data structures

→ Important improvement in the fight against the curse of dimensionality

→ Gives quantitative results regarding the need for higher resolutions
Training setup for 3D reconstruction

Partial Data \( X \)
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object \( Y \)

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

\[
L(D(E(X)), Y)
\]
Generating points PointSetGen[Fan2017]

Latent shape representation → MLP → Generated 3D points
Training setup for 3D reconstruction

Partial Data $X$
RGB Image(s)
RGBD Image(s)
PointCloud ...

Encoder $E$

Feature vector

Decoder $D$

Choice of representation
PointClouds (PointSetGen)

Loss ?

$L(D(E(X)), Y)$

3D Object $Y$
Loss on pointclouds

\[ L(\cdot, \cdot) = L(\cdot, \cdot) \]

<table>
<thead>
<tr>
<th>Complexity</th>
</tr>
</thead>
</table>
| EMD         | \( n^3 \)  
| Chamfer     | \( n^2 \)  

Thibault Groueix, Pierre-Alain Langlois, 2019
Loss on pointclouds

<table>
<thead>
<tr>
<th>Complexity</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EMD</td>
<td>$n^3$</td>
</tr>
<tr>
<td>Chamfer</td>
<td>$n^2$</td>
</tr>
</tbody>
</table>
Find the **optimal assignment** and compute

**Earth Mover Distance (EMD)**

- Hungarian Algorithm \([\text{Kuhn1955}] \sim O(n^3)\)
- Simplex based solver through LP formulation \(\sim O(\text{Hungarian})\)
- Sinkhorn regularization \([\text{Cuturi2013}]\) in near linear time \([\text{Altschuler2017}]\)
- \((1+\varepsilon)\) approximation \([\text{Bertsekas1988}]\) in \(\sim O(n^3)\)

\[
L((\begin{array}{c}
\text{Blue} \\
\text{Green} \\
\text{Gray}
\end{array}), (\begin{array}{c}
\text{Blue} \\
\text{Green} \\
\text{Gray}
\end{array})) = L((\begin{array}{c}
\text{Blue} \\
\text{Green} \\
\text{Gray}
\end{array}), (\begin{array}{c}
\text{Blue} \\
\text{Green} \\
\text{Gray}
\end{array})) = \frac{1}{5} \cdot (d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2)
\]

<table>
<thead>
<tr>
<th></th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMD</td>
<td>(n^3)</td>
</tr>
<tr>
<td>Chamfer</td>
<td>(n^2)</td>
</tr>
</tbody>
</table>
Loss on pointclouds

Find the nearest neighbours and compute Chamfer Distance (CD) = L(○, ●) +

\[ L(\text{○}, \text{●}) = L(\text{●}, \text{○}) = \frac{1}{5} \cdot (d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2) \]

<table>
<thead>
<tr>
<th></th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMD</td>
<td>n³</td>
</tr>
<tr>
<td>Chamfer</td>
<td>n²</td>
</tr>
</tbody>
</table>

Thibault Groueix, Pierre-Alain Langlois, 2019
Loss on pointclouds

Find the nearest neighbours and compute

\[ \text{Chamfer Distance (CD)} = L(\bullet, \circ) + \]

\[ L(\bullet, \circ) = L(\bullet, \circ) \]

\[ = \frac{1}{5} \cdot (d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2) \]

<table>
<thead>
<tr>
<th></th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMD</td>
<td>(n^3)</td>
</tr>
<tr>
<td>Chamfer</td>
<td>(n^2)</td>
</tr>
</tbody>
</table>
Loss on pointclouds

Find the nearest neighbours and compute

**Chamfer Distance (CD)** = \( L(\bullet, \bullet) + L(\bullet, \bullet) \)

\[
L(\bullet, \bullet) = L(\bullet, \bullet) = \frac{1}{5} \cdot (d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2)
\]

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMD</td>
<td>(n^3)</td>
</tr>
<tr>
<td>Chamfer</td>
<td>(n^2)</td>
</tr>
</tbody>
</table>
Loss on pointclouds: the mean shape carries characteristics of the distance metric

\[ \bar{x} = \arg \min_x \mathbb{E}_{s \sim \mathcal{S}}[d(x, s)] \]

Credit: [Fan2016]
Training setup for 3D reconstruction

Partial Data \( X \)
- RGB Image(s)
- RGBD Image(s)
- PointCloud

3D Object \( Y \)

Encoder \( E \)
Decoder \( D \)
Choice of representation
PointClouds (PointSetGen)

Loss:
Chamfer Distance

\[ L(D(E(X)), Y) \]
Generating points

Test Shape
Generating points

Test Shape

Latent shape representation

Generated 3D points

Encoder

Decoder
Generating points

Test Shape

Latent shape representation

MLP

Generated 3D points
Limitation of PointSetGen [Fan2017]

- Generate a fixed number of points
- Points connectivity is missing
- Generated points are not correlated enough to belong to an implicit surface
Limitation of PointSetGen [Fan2017]

➔ Generate a fixed number of points
➔ **Points connectivity is missing**
➔ Generated points are not correlated enough to belong to an implicit surface
Limitation of PointSetGen [Fan2017]

➔ Generate a fixed number of points
➔ **Points connectivity is missing**
➔ Generated points are not correlated enough to belong to an implicit surface
Limitation of PointSetGen [Fan2017]

➔ Generate a fixed number of points
➔ Points connectivity is missing
➔ Generated points are not correlated enough to belong to an implicit surface
Limitation of PointSetGen [Fan2017]

- Generate a fixed number of points
- **Points connectivity is missing**
- Generated points are not correlated enough to belong to an implicit surface
Limitation of PointSetGen [Fan2017]

➔ Generate a fixed number of points
➔ Points connectivity is missing
➔ Generated points are not correlated enough to belong to an implicit surface

Reconstructing the mesh from a pointcloud:
Poisson Surface Reconstruction [Kazhdan2013]
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

Encoder $E$

Decoder $D$

Feature vector

Loss $L(D(E(X)), Y)$
Deform a surface [Groueix2018]
Deform a surface: space mapping trick [Groueix2018]

Latent shape representation
Sampled 2D point
MLP
Generated 3D point
Deform a surface \([\text{Groueix2018}]\)
Deform a surface [Groueix2018]
Deform a surface [Groueix2018]
Deform a surface [Groueix2018]
Deform a surface [Groueix2018]
Deform a surface [Groueix2018]

Test Shape

- Sampled 2D point
- Latent shape representation
- MLP 2
- Generated 3D point

Encoder (E) -> Latent shape representation -> MLP 2 -> Generated 3D point

Decoder (D)

- Test Shape

Deform a surface [Groueix2018]
Deform a surface [Groueix2018]

Test Shape

Latent shape representation

Sampled 2D point

MLP 2

Generated 3D point

Encoder E

Decoder D

Sampled 2D point

Latent shape representation

MLP 2

Generated 3D point

Test Shape
Deform a surface [Groueix2018]

Test Shape

Encoder E

Decoder D

Latent shape representation

Sampled 2D point

MLP 1

Generated 3D point

Test Shape
Deform a surface [Groueix2018]

Test Shape

Latent shape representation

Sampled 2D point

MLP 3

MLP 2

MLP 1

Generated 3D point
Deform a surface [Groueix2018]

Test Shape

Latent shape representation

Sampled 2D point

MLP 1

MLP 2

MLP 3

Generated 3D point

Test Shape

Encoder

Decoder
Results: Single View Reconstruction

(a) Input  (b) 3D-R2N2  (c) HSP  (d) PSG  (e) Ours
Direct application: mesh parametrization
State-of-the-art correspondences of FAUST [Groueix2018b]
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Encoder $E$

Decoder $D$

Feature vector

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

$L(D(E(X)), Y)$

Loss
Can the space mapping trick be applied to volumetric representations?

-> yes, through the Signed Distance Function (SDF) ! [Mescheder2018], [Park2019], [Chen2019]
Deform a surface: space mapping trick [Groueix2018]

Latent shape representation

Sampled 2D point P

Generated 3D point on the surface MLP(P)

Supervise with the Chamfer Distance

MLP
Deform a volume [Mescheder2018]

Latent shape representation

Sampled 3D point P

Predict Signed Distance Function : SDF(P)

Supervise with ?

MLP

SDF(P)
Deform a volume [Mescheder2018]

Sampled 3D point $P$

Predict Signed Distance Function: $\text{SDF}(P)$

Latent shape representation

Supervise with:
- Classification: cross-entropy
- Regression: L1 Loss

Decoder $D$
From the SDF to a mesh: marching cubes [Liao2018, Lorensen1987]

Core idea: the surface of the object corresponds to the 0-level set of the SDF.

Credit: [Park2019]
Can the space mapping trick be applied on volumes ?

-> yes, through the Signed Distance Function (SDF) ! [Mescheder2018], [Park2019], [Chen2019]

++ Get a voxel based representation at infinite granularity
++ Get analytic normals : $d\text{SDF}(x)/dx$
++ Topology is no longer an issue
-- need only one assumption : there is an interior and an exterior
Single View Reconstruction Results [Mescheder2018]
Test on Real Images [Mescheder2018]
Interpolation Results [Park2019]
Limitations so far - SDF

Reconstructed models are too smooth
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point $p$ uniformly and compute its sdf $r$:

- If $r > 0$, the circle $(p, r)$ is **full**
- If $r < 0$, the circle $(p, -r)$ is **empty**
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point $p$ uniformly and compute its sdf $r$:

- If $r > 0$, the circle $(p, r)$ is full
- If $r < 0$, the circle $(p, -r)$ is empty
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point \( p \) uniformly and compute its sdf \( r \):

- If \( r > 0 \), the circle \((p, r)\) is full
- If \( r < 0 \), the circle \((p, -r)\) is empty
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation : Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point \( p \) uniformly and compute its sdf \( r \):

- If \( r > 0 \), the circle \( (p, r) \) is full
- If \( r < 0 \), the circle \( (p, -r) \) is empty
Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point $p$ uniformly and compute its sdf $r$:

- If $r > 0$, the circle $(p, r)$ is **full**
- If $r < 0$, the circle $(p, -r)$ is **empty**
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point $p$ uniformly and compute its sdf $r$:

- If $r > 0$, the circle $(p, r)$ is full
- If $r < 0$, the circle $(p, -r)$ is empty
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point \( p \) uniformly and compute its sdf \( r \):

- If \( r > 0 \), the circle \( (p, r) \) is **full**
- If \( r < 0 \), the circle \( (p, -r) \) is **empty**
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point $p$ uniformly and compute its sdf $r$:

- If $r > 0$, the circle $(p, r)$ is full
- If $r < 0$, the circle $(p, -r)$ is empty

Number of samples: 100
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point \( p \) uniformly and compute its sdf \( r \):

- If \( r > 0 \), the circle \( (p, r) \) is full
- If \( r < 0 \), the circle \( (p, -r) \) is empty
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point $p$ uniformly and compute its sdf $r$:

- If $r > 0$, the circle $(p, r)$ is full
- If $r < 0$, the circle $(p, -r)$ is empty

Number of samples: 500
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point $p$ uniformly and compute its SDF $r$:

- If $r > 0$, the circle $(p, r)$ is full
- If $r < 0$, the circle $(p, -r)$ is empty

Number of samples: 1000
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point p uniformly and compute its sdf r:

- If $r > 0$, the circle $(p, r)$ is full
- If $r < 0$, the circle $(p, -r)$ is empty

Little information at the interior of sharp areas -> no supervision -> bad predictions
Limitations so far - SDF

Reconstructed models are too smooth - Possible explanation: Monte-Carlo sampling

We want to approximate the orange square’s SDF through Monte-Carlo. We draw a point \( p \) uniformly and compute its sdf \( r \) :

- If \( r > 0 \), the circle \((p, r)\) is full
- If \( r < 0 \), the circle \((p, -r)\) is empty

Little information at the interior of sharp areas -> no supervision -> bad predictions

Potential fix: non uniform sampling
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)
...

3D Object $Y$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

$L(D(E(X)), Y)$
Fitting geometric primitives to a 3D shape

Everything in nature takes its form from the sphere, the cone and the cylinder. - Paul Cezanne.

Motivations:

● Parsimony of description

● Helps finding structures in images for abstraction or animation

● In the case of geometric object, helps capturing details (sharp angles)
Learn 3D reconstruction with cuboids [Tulsiani2017]

Unsupervised method for fitting cuboid primitives

Latent representation

Primitive parameters (position, scale, existence)

Loss
Learn 3D reconstruction with cuboids [Tulsiani2017]

Unsupervised method for fitting cuboid primitives

Challenges:

1. Position the cuboids
2. Estimating the amount of cuboids to predict
Fitting cuboids to a 3D shape \cite{Tulsiani2017} 

Designing a loss: Chamfer? 

Problem!

$L(\bullet, \bigcirc)$: $\subseteq$ $\subseteq$
Fitting cuboids to a 3D shape \cite{Tulsiani2017}

Designing a loss: Chamfer? $L(\cdot, \bigcirc) : \bigcirc \subset \square$

Problem!

→ Points sampled on increase the Chamfer distance
Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss: Chamfer?

$L(\square, \circ)$

Problem!

→ Points sampled on $\square$ increase the Chamfer distance

→ Solution:
  - Among points sampled on $\circ$, we discard points which are inside $\square$
  - Among points sampled on $\square$, we discard points which are inside $\circ$
Estimating the amount of cuboids to predict [Tulsiani2017]

We don’t know whether a predicted primitive exists or not (unsupervised setting).

How to efficiently learn it?
Estimating the amount of cuboids to predict [Tulsiani2017]

We don’t know whether a predicted primitive exists or not (unsupervised setting).

How to efficiently learn it?

For each predicted primitive, we define a Bernoulli random variable $z_m$ with parameter $\Theta_m$. 
Estimating the amount of cuboids to predict \cite{Tulsiani2017}

We don’t know whether a predicted primitive exists or not (unsupervised setting).

How to efficiently learn it?

For each predicted primitive, we define a Bernoulli random variable $z_m$ with parameter $\Theta_m$

$L(\bigcup_m (\bar{P}_m, z_m), O)$ is a version of the loss which just ignores the $m$-th primitive when $z_m = 0$
Estimating the amount of cuboids to predict [Tulsiani2017]

We don’t know whether a predicted primitive exists or not (unsupervised setting).

How to efficiently learn it?

For each predicted primitive, we define a Bernoulli random variable $z_m$ with parameter $\theta_m$.

$L(\cup_m(\bar{P}_m, z_m), O)$ is a version of the loss which just ignores the $m$-th primitive when $z_m = 0$.

**Final loss:** $L_{fin}(\{(\bar{P}_m, p_m), \forall m\}, O) = \mathbb{E}_{\forall m, z_m \sim \text{Bern}(p_m)} L(\cup_m(\bar{P}_m, z_m), O)$
Estimating the amount of cuboids to predict \[\text{[Tulsiani2017]}\]

We don’t know whether a predicted primitive exists or not (unsupervised setting).

How to efficiently learn it?

For each predicted primitive, we define a Bernoulli random variable \(z_m\) with parameter \(\theta_m\).

\(L(\bigcup_m (\bar{P}_m, z_m), O)\) is a version of the loss which just ignores the \(m\)-th primitive when \(z_m = 0\).

Final loss: \(L_{fin}(\{(\bar{P}_m, p_m), \forall m\}, O) = \mathbb{E}_{z_m \sim \text{Bern}(p_m)} L(\bigcup_m (\bar{P}_m, z_m), O)\)

“Average loss that we get when choosing the primitive existence w.r.t the parameters \(\theta_m\)”
Estimating the amount of cuboids to predict \cite{Tulsiani2017}

We don’t know whether a predicted primitive exists or not (unsupervised setting).

How to efficiently learn it?

For each predicted primitive, we define a Bernoulli random variable $z_m$ with parameter $\theta_m$

$L(\cup_m (\bar{P}_m, z_m), O)$ is a version of the loss which just ignores the $m$-th primitive when $z_m = 0$

Final loss: $L_{fin}(\{ (\bar{P}_m, p_m), \forall m \}, O) = \mathbb{E}_{\forall m, z_m \sim Bern(p_m)} L(\cup_m (\bar{P}_m, z_m), O)$

“Average loss that we get when choosing the primitive existence w.r.t the parameters $\theta_m$”

How to back-propagate through an expectation?
Estimating the amount of cuboids to predict [Tulsiani2017]

Back-propagate through an expectation [Williams1992]

Let $X : \Omega \rightarrow \mathcal{X}$ be a discrete random variable with p.d.f $p_{\theta}$ parametrized by $\theta$.
Let $f : \mathcal{X} \rightarrow \mathbb{R}$
We want to evaluate
Estimating the amount of cuboids to predict \cite{Tulsiani2017}

Back-propagate through an expectation \cite{Williams1992}

Let $X : \Omega \rightarrow \mathcal{X}$ be a discrete random variable with p.d.f $p_\theta$ parametrized by $\theta$.
Let $f : \mathcal{X} \rightarrow \mathbb{R}$
We want to evaluate

$$\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_\theta(x)$$
Estimating the amount of cuboids to predict [Tulsiani2017]

Back-propagate through an expectation [Williams1992]

Let $X : \Omega \rightarrow \mathcal{X}$ be a discrete random variable with p.d.f $p_\theta$ parametrized by $\theta$.
Let $f : \mathcal{X} \rightarrow \mathbb{R}$
We want to evaluate

$$\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_\theta(x)$$

$$= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} p_\theta(x)$$
Estimating the amount of cuboids to predict [Tulsiani2017]

Back-propagate through an expectation [Williams1992]

Let $X : \Omega \to \mathcal{X}$ be a discrete random variable with p.d.f $p_{\theta}$ parametrized by $\theta$.
Let $f : \mathcal{X} \to \mathbb{R}$
We want to evaluate

$$\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x) p_{\theta}(x)$$

$$= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} p_{\theta}(x)$$

$$= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} \log(p_{\theta}(x)) p_{\theta}(x) \quad \text{"log-likelihood trick"}$$
Estimating the amount of cuboids to predict [Tulsiani2017]

Back-propagate through an expectation [Williams1992]

Let $X : \Omega \to \mathcal{X}$ be a discrete random variable with p.d.f $p_\theta$ parametrized by $\theta$.
Let $f : \mathcal{X} \to \mathbb{R}$
We want to evaluate

$$\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_\theta(x)$$

$$= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} p_\theta(x)$$

$$= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} \log(p_\theta(x))p_\theta(x) \quad \text{“log-likelihood trick”}$$

$$= \mathbb{E} \left[ f(X) \frac{\partial}{\partial \theta} \log(p_\theta(X)) \right]$$
Estimating the amount of cuboids to predict \cite{Tulsiani2017}

Back-propagate through an expectation \cite{Williams1992}

Let $X : \Omega \to \mathcal{X}$ be a discrete random variable with p.d.f $p_\theta$ parametrized by $\theta$.
Let $f : \mathcal{X} \to \mathbb{R}$
We want to evaluate

$$
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_\theta(x)
$$

$$
= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} p_\theta(x)
$$

$$
= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} \log(p_\theta(x))p_\theta(x) \quad \text{“log-likelihood trick”}
$$

$$
= \mathbb{E} \left[ f(X) \frac{\partial}{\partial \theta} \log(p_\theta(X)) \right]
$$

The expectation can be estimated thanks to Monte Carlo with $(X_n)_{n \in \{1, N\}} \sim p_\theta$
Estimating the amount of cuboids to predict [Tulsiani2017]

Back-propagate through an expectation [Williams1992]

Let $X : \Omega \to \mathcal{X}$ be a discrete random variable with p.d.f $p_{\theta}$ parametrized by $\theta$.
Let $f : \mathcal{X} \to \mathbb{R}$
We want to evaluate

$$
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_{\theta}(x)
$$

$$
= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} p_{\theta}(x)
$$

$$
= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} \log(p_{\theta}(x))p_{\theta}(x)
$$

“log-likelihood trick”

$$
= \mathbb{E} \left[ f(X) \frac{\partial}{\partial \theta} \log(p_{\theta}(X)) \right]
$$

The expectation can be estimated thanks to Monte Carlo with $(X_n)_{n \in \{1, N\}} \sim p_{\theta}$

$$
\approx \frac{1}{N} \sum_{n=1}^{N} \left[ f(X_n) \frac{\partial}{\partial \theta} \log(p_{\theta}(X_n)) \right]
$$
Estimating the amount of cuboids to predict [Tulsiani2017]

Back-propagate through an expectation [Williams1992]

Let $X : \Omega \to \mathcal{X}$ be a discrete random variable with p.d.f $p_{\theta}$ parametrized by $\theta$.
Let $f : \mathcal{X} \to \mathbb{R}$

$$
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] \approx \frac{1}{N} \sum_{n=1}^{N} \left[ f(X_n) \frac{\partial}{\partial \theta} \log(p_{\theta}(X_n)) \right]
$$

Monte-Carlo sampling

This approximation is good when the dimension of $\mathcal{X}$ is not too high.
Learn 3D reconstruction with cuboids [Tulsiani2017]

Results

Notice that different shapes are reconstructed with different sets of cuboids
Learn 3D reconstruction with superquadrics [Paschalidou2019]

*Everything in nature takes its form from the sphere, the cone and the cylinder.*
- Paul Cezanne.
Learn 3D reconstruction with superquadrics \[\text{[Paschalidou2019]}\]

Results

\[\text{[Tulsiani2017]}\]

\[\text{[Paschalidou2019]}\]
Learn 3D reconstruction with superquadrics [Paschalidou2019]

Results

+ Generality
+ Parsimony of description
+ Inter-object coherence for signal transfer
- Data fidelity
- Training Stability

[Tulsiani2017]  

[Passchalidou2019]
Learn 3D reconstruction with superquadrics [Paschalidou2019]

Results

+ Generality
+ Parsimony of description
+ Inter-object coherence for signal transfer

- Data fidelity
- Training Stability

[Tulsiani2017]

[Paschalidou2019]
Learn 3D reconstruction with superquadrics \cite{Paschalidou2019}

\[
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in X} f(x)p_\theta(x)
\]

(b) Evolution of Training Loss.
Learn 3D reconstruction with superquadrics [Paschalidou2019]

\[
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_\theta(x)
\]

Are there alternatives to REINFORCE?

- The Reparameterization trick: cf [MohamedSlides]
- Direct analytical computation in the particular case of the Chamfer Distance
Learn 3D reconstruction with superquadrics [Paschalidou2019]

\[
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in X} f(x)p_\theta(x)
\]

Are there alternatives to REINFORCE?

- The Reparameterization trick: cf [MohamedSlides]
- Direct analytical computation in the particular case of the Chamfer Distance
\[
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_{\theta}(x)
\]

\[
\mathbb{E}[f(X)] = \mathbb{E}_{p(\theta)} \left[ \left( \sum_{m=1}^{M} \mathcal{L}(\tilde{P}_m, X) \right) + \mathcal{L}(X, \tilde{P}) \right]
\]

\[
\mathcal{L}(A, B) = \sum_{x \in A} \Delta(x, B)
\]

\[
\Delta(x, B) = \min_{y \in B} ||x - y||
\]
\[
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_\theta(x)
\]

\[
\mathbb{E}[f(X)] = \mathbb{E}_{p(\theta)} \left[ \left( \sum_{m=1}^{M} \mathcal{L}(\tilde{\mathbf{P}}_m, \mathbf{X}) \right) + \mathcal{L}(\mathbf{X}, \tilde{\mathbf{P}}) \right]
\]

\[
= \sum_{m=1}^{M} \theta_m \mathcal{L}(\mathbf{P}_m, \mathbf{X}) + \mathbb{E}_{p(\theta)} \left[ \sum_{x_i \in \mathbf{X}} \min_{m: z_m=1} \Delta(x_i, \mathbf{P}_m) \right]
\]

\[
= \sum_{m=1}^{M} \theta_m \mathcal{L}(\mathbf{P}_m, \mathbf{X}) + \mathbb{E}_{p(\theta)} \left[ \sum_{x_i \in \mathbf{X}} \min_{m: z_m=1} \Delta_i^m \mathbf{P}_m \right]
\]

\[
\mathcal{L}(\mathbf{A}, \mathbf{B}) = \sum_{\mathbf{x} \in \mathbf{A}} \Delta(\mathbf{x}, \mathbf{B})
\]

\[
\Delta(\mathbf{x}, \mathbf{B}) = \min_{y \in \mathbf{B}} \|x - y\|
\]

\[\text{2}^M \text{ configurations}\]
\[
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x)p_\theta(x)
\]

\[
\mathbb{E}[f(X)] = \mathbb{E}_{p(\theta)} \left[ \left( \sum_{m=1}^{M} \mathcal{L}(\tilde{P}_m, X) \right) + \mathcal{L}(X, \tilde{P}) \right]
\]

\[
= \sum_{m=1}^{M} \theta_m \mathcal{L}(P_m, X) + \mathbb{E}_{p(\theta)} \left[ \sum_{x_i \in X} \min_{m|z_m=1} \Delta(x_i, P_m) \right]
\]

\[
= \sum_{m=1}^{M} \theta_m \mathcal{L}(P_m, X) + \mathbb{E}_{p(\theta)} \left[ \sum_{x_i \in X} \min_{m|z_m=1} \Delta_i^m, P_m \right]
\]

\[
\Delta_i^1 \leq \Delta_i^2 \leq \cdots \leq \Delta_i^M
\]

\[
\min_{m|z_m=1} \Delta_i^m = \begin{cases} 
\Delta_i^1, & \text{if } z_1 = 1 \\
\Delta_i^2, & \text{if } z_1 = 0, z_2 = 1 \\
\vdots & \\
\Delta_i^M, & \text{if } z_m = 0, \ldots, z_M = 1
\end{cases}
\]

\[
\Delta(x, B) = \min_{y \in B} ||x - y||
\]

\[
\mathcal{L}(A, B) = \sum_{x \in A} \Delta(x, B)
\]

2^M configurations

Reordering trick
\[ \frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x) p_\theta(x) \]

\[ \mathbb{E}[f(X)] = \mathbb{E}_{p(\theta)} \left[ \left( \sum_{m=1}^{M} \mathcal{L}(\tilde{P}_m, X) \right) + \mathcal{L}(X, \tilde{P}) \right] \]

\[ = \sum_{m=1}^{M} \theta_m \mathcal{L}(P_m, X) + \mathbb{E}_{p(\theta)} \left[ \sum_{m|z_m=1} \min_{x_i \in X} \Delta(x_i, P_m) \right] \]

\[ = \sum_{m=1}^{M} \theta_m \mathcal{L}(P_m, X) + \mathbb{E}_{p(\theta)} \left[ \sum_{m|z_m=1} \min_{x_i \in X} \Delta_i^m, P_m \right] \]

\[ \Delta_i^1 \leq \Delta_i^2 \leq \cdots \leq \Delta_i^M \]

\[ \min_{m|z_m=1} \Delta_i^m = \begin{cases} 
\Delta_i^1, & \text{if } z_1 = 1 \\
\Delta_i^2, & \text{if } z_1 = 0, z_2 = 1 \\
\vdots \\
\Delta_i^M, & \text{if } z_m = 0, \ldots, z_M = 1 
\end{cases} \]

\[ \mathbb{E}[f(X)] = \sum_{m=1}^{M} \theta_m \mathcal{L}(P_m, X) + \sum_{x_i \in X} \sum_{m=1}^{M} \Delta_i^m \theta_m \prod_{m=1}^{m-1} (1 - \theta_m) \]

\[ \mathcal{L}(A, B) = \sum_{x \in A} \Delta(x, B) \]

\[ \Delta(x, B) = \min_{y \in B} ||x - y|| \]

Complexity: \(2^M\) configurations

Reordering trick

Complexity: \(2^M \rightarrow M^2\)
Learn 3D reconstruction with superquadrics [Paschalidou2019]

\[
\frac{\partial}{\partial \theta} \mathbb{E}[f(X)] = \frac{\partial}{\partial \theta} \sum_{x \in X} f(x) p_{\theta}(x)
\]

Are there alternatives to REINFORCE?

- The Reparameterization trick: cf [MohamedSlides]
- Direct analytical computation in the particular case of the Chamfer Distance

(b) Evolution of Training Loss.
Partial Data $X$
RGB Image(s)
RGBD Image(s)
PointCloud
PointCloud (Voxelized)
...

3D Object $Y$

Training setup for 3D reconstruction

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

Feature vector

$L(D(E(X)), Y)$

Loss

Thibault Groueix, Pierre-Alain Langlois, 2019
Training setup for 3D reconstruction

Partial Data $\mathbf{X}$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $\mathbf{Y}$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

Feature vector

Encoder $\mathbf{E}$

Decoder $\mathbf{D}$

Loss $L(D(E(\mathbf{X})), \mathbf{Y})$
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

$\rightarrow$ Many image dataset, but scarcity of 3D models
$\rightarrow$ 3D-based losses fail to capture sharp angles
$\rightarrow$ Hybrid approaches between learning and optimization

3D Object $Y$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

$\mathbb{L}(D(E(X)), Y)$
Training setup for 3D reconstruction

Partial Data $\mathbf{X}$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

Encoder $E$
Decoder $D$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

$L(D(E(\mathbf{X})), \mathbf{Y})$

→ Many image dataset, but scarcity of 3D models
→ 3D-based losses fail to capture sharp angles
→ Hybrid approaches between learning and optimization
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)
...

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

Encoder $E$
- Feature vector

Decoder $D$

→ Many image dataset, but scarcity of 3D models
→ 3D-based losses fail to capture sharp angles
→ Hybrid approaches between learning and optimization

Loss

Rendering

$L(D(E(X)), Y)$
Inverse Rendering - Issues

➔ Rasterization is not differentiable
Inverse Rendering - Issues

→ Z-buffering is not differentiable
Differentiating the rasterization process - silhouette case

[Kato2018]

→ Avoid the z-buffering process by just rendering silhouettes
Differentiating the rasterization process - silhouette case

[Kato2018]

→ Get a differentiable process through blurring

Figure 2. Illustration of our method. \( v_i = \{x_i, y_i\} \) is one vertex of the face. \( I_j \) is the color of pixel \( P_j \). The current position of \( x_i \) is \( x_0 \), \( x_1 \) is the location of \( x_i \) where an edge of the face collides with the center of \( P_j \) when \( x_i \) moves to the right. \( I_j \) becomes \( I_{ij} \) when \( x_i = x_1 \).

Figure 3. Illustration of our method in the case where \( P_j \) is inside the face. \( I_j \) changes when \( x_i \) moves to the right or left.
Differentiating the rasterization process - silhouette case

[Kato2018]

→ Direct application, render a sphere and optimize its rendering to the silhouette of an input image
Differentiating the rasterization process - silhouette case

[Kato2018]

➔ Direct application, render a sphere and optimize its rendering to the silhouette of an input image

Problem:

➔ Requires strong regularization to work!

Figure 5. Generation of the back side of a CRT monitor with/without smoothness regularizer. Left: input image. Center: prediction without regularizer. Right: prediction with regularizer.
Differentiating the rasterization process - silhouette case

[Kato2018]

Problem:

➔ Requires strong regularization to work!

Ideas:

➔ Use multiple views [Petersen2019]

➔ Improve the rendering process
  [Nguyen-Phuoc2018], [Petersen2019], [Yang2018]
Training setup for 3D reconstruction

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

Loss
$L(D(E(X)), Y)$
Very Modular Framework! [Tulsiani2017]

Partial Data $\mathbf{X}$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $\mathbf{Y}$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

$L(D(E(\mathbf{X})), \mathbf{Y})$
Very Modular Framework! [Groueix2018]

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

Feature vector

Encoder $E$

Decoder $D$

Loss $L(D(E(X)), Y)$

$L(D(E(X)), Y)$

Very Modular Framework! [Groueix2018]

$E(D(E(X)), Y)$
Very Modular Framework! [Choy2016, Tatarchenko2017]

Partial Data $X$
- RGB Image(s)
- RGBD Image(s)
- PointCloud
- PointCloud (Voxelized)

3D Object $Y$

Choice of representation
- Volumetric (OctNet)
- PointClouds (PointSetGen)
- Surfaces (AtlasNet)
- Signed Distance Function
- Geometric Primitives

$L(D(E(X)), Y)$

Very Modular Framework!
Limitations of learned approaches

- Hard to add geometric constraints in the design of a neural net architecture e.g. Watertight reconstruction. cf http://imagine.enpc.fr/~groueixt/atlasnet/viewer-svr/

- Hard to scale to large scenes and/or very high level of details.

- Biased by data

- ...
What was not covered today

**Traditional methods**: Shape from X

**Graph Based methods**: Spectral and spatial methods

**Equivariant methods**: Spherical CNNs

**Other Point Based Methods**: PCPNet, Kd-Trees

**Differential rendering for inverse graphics**: Neural renderer, rendernet

**2.5D and Layer-Structured Inference**: [Tulsiani2018]

**Making it work on real sensor data**: domain adaptation, data augmentation
Take Home Message

The choice of representation of 3D data is critical

We journeyed from *Volumes*..., 
... through *Pointclouds*..., 
to *Surfaces*.

Thank you
Bibliography: Encoder

Points


Spherical representations


Graph

Voxels

Images
Points

Voxels

Surfaces

Mesh

Depth maps

Signed Distance Function
Bibliography : Decoder

Geometric primitives


Optimal Transport


Datasets


Marching Cubes


Bibliography

Other


