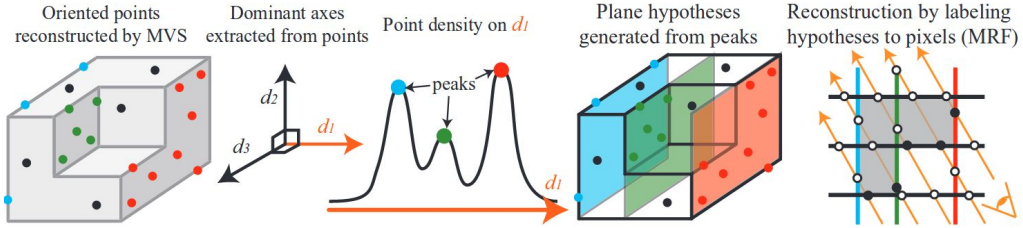
Deep Learning for 3D Toward surface generation

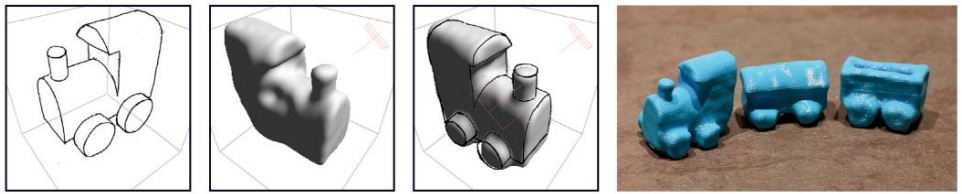
Thibault GROUEIX, Pierre-Alain LANGLOIS

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/sensors are relevant as input for 3d reconstruction ?

RGB Image(s)

RGBD Image(s)

PointCloud

Typical learning framework based on synthetic data

Partial Data **X**

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**

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

Feature vector

Decoder D

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
?

Encoder
E

Feature vector

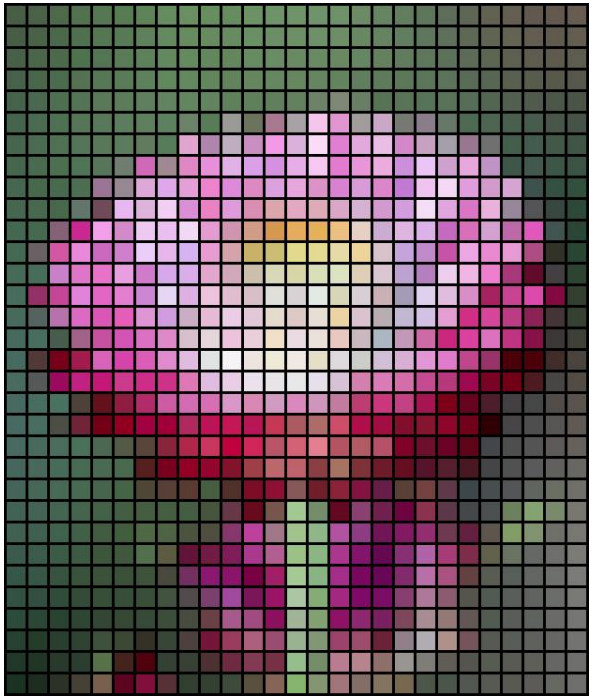
Decoder
D

Loss

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

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)

...

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)

Training setup for 3D reconstruction

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

PointCloud (Voxelized)

...

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)

3D Object **Y**

Training setup for 3D reconstruction

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

PointCloud (Voxelized)

...

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)

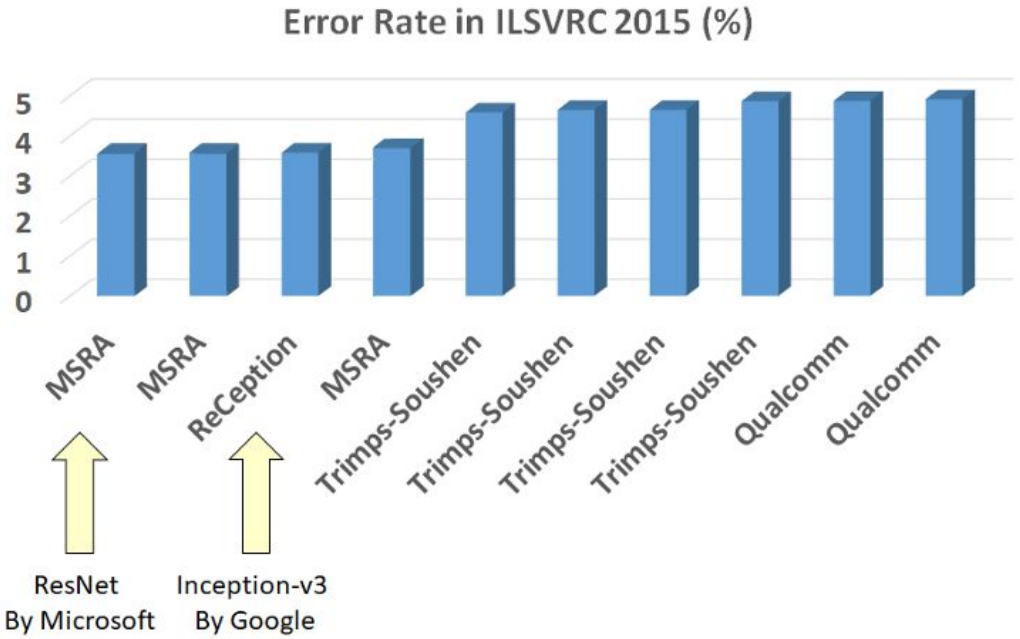
3D Object **Y**

Encoder

E

Encoders for RGB & RGBD images

Do not reinvent the wheel :
Use State-of-the-art 2D networks



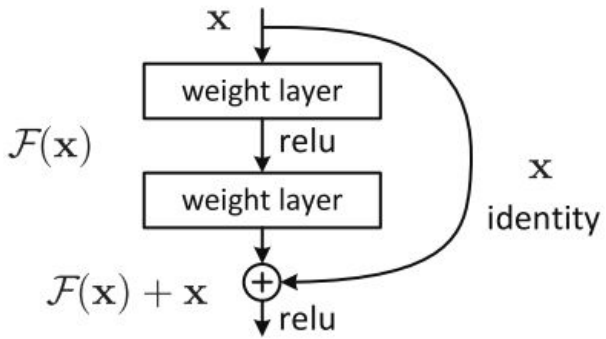
Encoder

E

Encoders for RGB & RGBD images

Do not reinvent the wheel :
Use State-of-the-art 2D networks

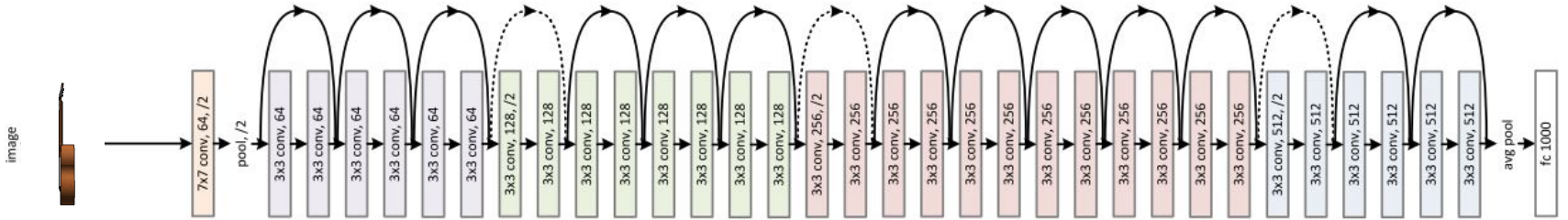
- Resnet [He2015] -> Skip connections
- BatchNorm [Ioffe2015]



Resnet 34 [He2015]

Encoder

E



Training setup for 3D reconstruction

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

PointCloud (Voxelized)

...

Encoder **E**

Feature vector

Decoder **D**

Choice of representation

Volumetric (OctNet)

PointClouds (PointSetGen)

Surfaces (AtlasNet)

Signed Distance Function

Geometric Primitives

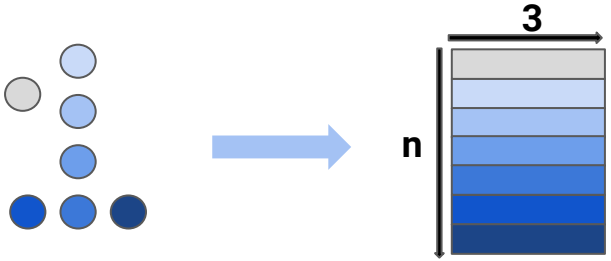
Loss

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

3D Object **Y**

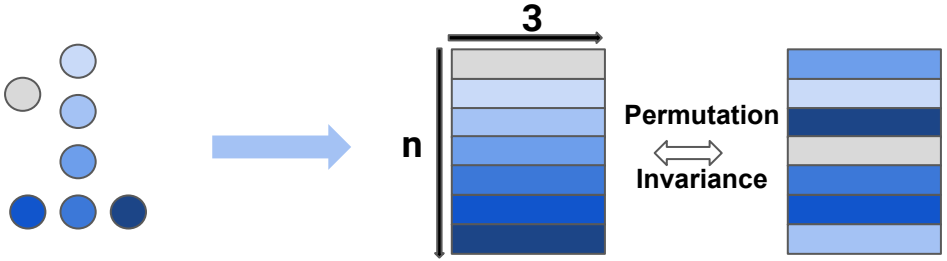
Encoder

E



Encoder

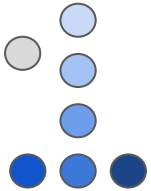
E

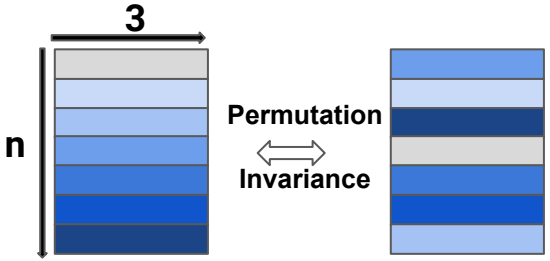


PointNet [Qi2017]

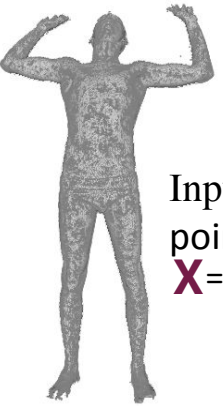
Encoder

E





PointNet [Qi2017]

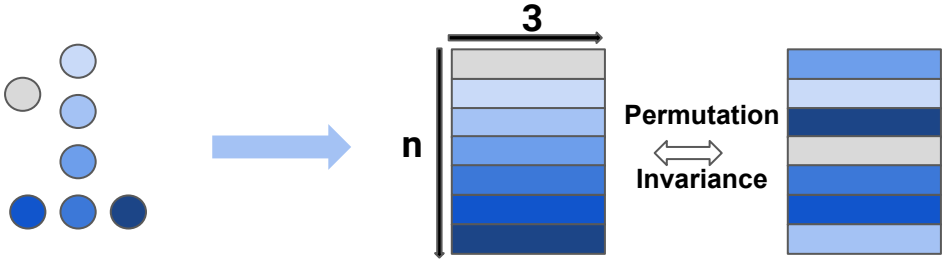


Input pointcloud

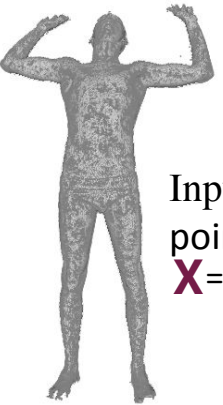
$\mathbf{X} = (x_1, x_2, \ldots, x_n)$

Encoder

E



PointNet [Qi2017]



Input pointcloud

$\mathbf{X} = (x_1, x_2, \ldots, x_n)$

$x_1 = (1,2,3) \rightarrow$

$x_2 = (1,1,1) \rightarrow$

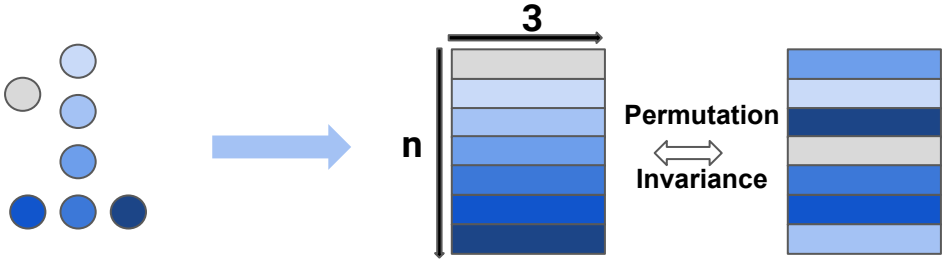
$x_{\ldots} = (2,3,2) \rightarrow$

$x_n = (2,3,4) \rightarrow$

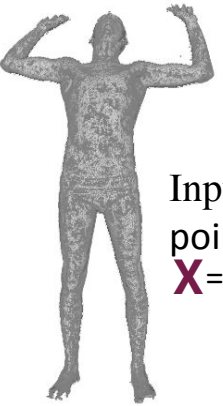
$$\mathbf{E}((x_1, x_2, \ldots, x_n)) = \qquad x_1 \ , \ \ldots, \qquad x_n$$

Encoder

E

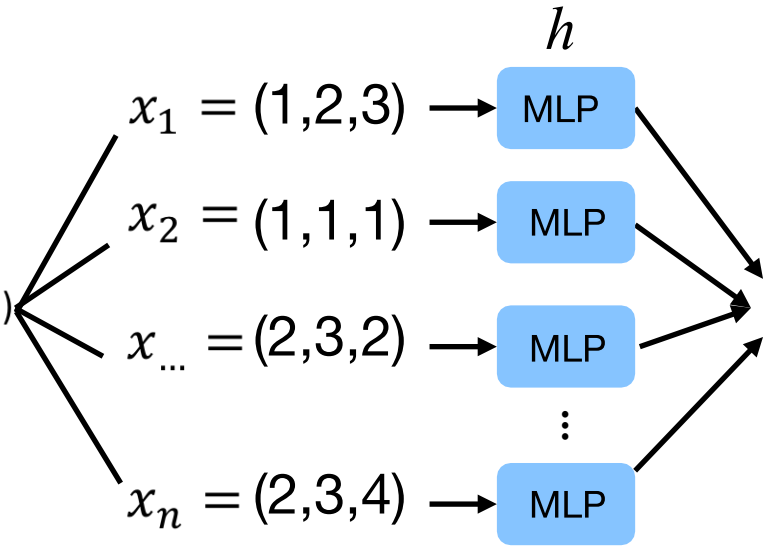


PointNet [Qi2017]



Input pointcloud

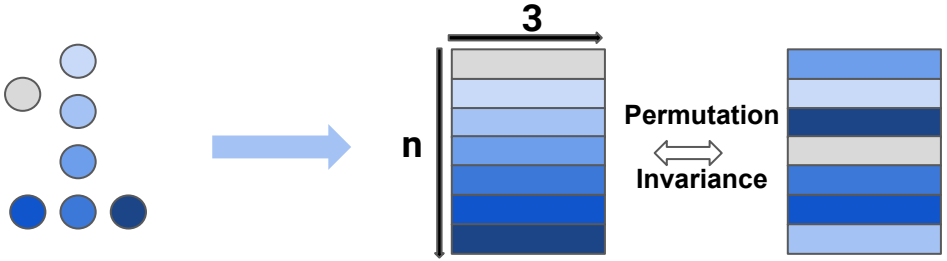
$\mathbf{X} = (x_1, x_2, \ldots, x_n)$



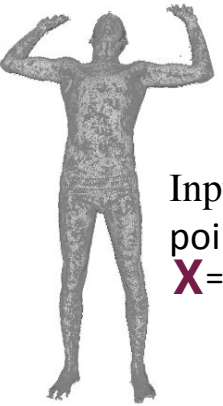
$$\mathbf{E}((x_1, x_2, \ldots, x_n)) = \quad h(x_1), \ldots, h(x_n)$$

Encoder

E

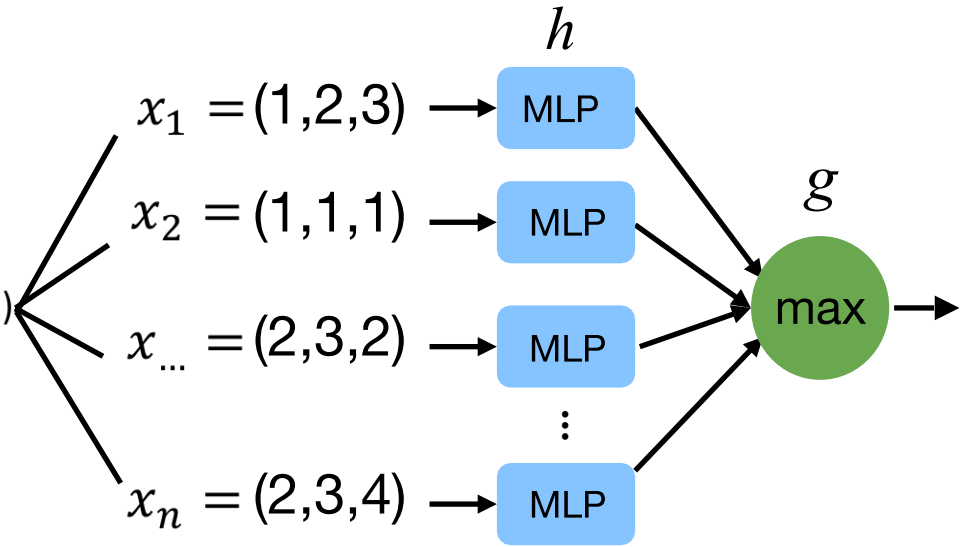


PointNet [Qi2017]



Input pointcloud

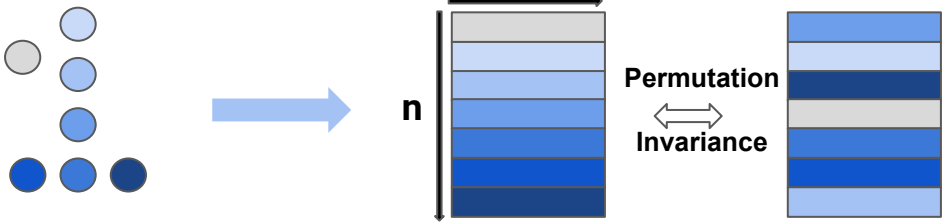
$\mathbf{X} = (x_1, x_2, \ldots, x_n)$



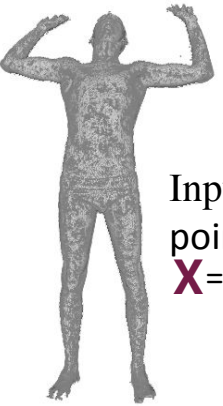
$$\mathbf{E}((x_1, x_2, \ldots, x_n)) = g(h(x_1), \ldots, h(x_n))$$

Encoder

E

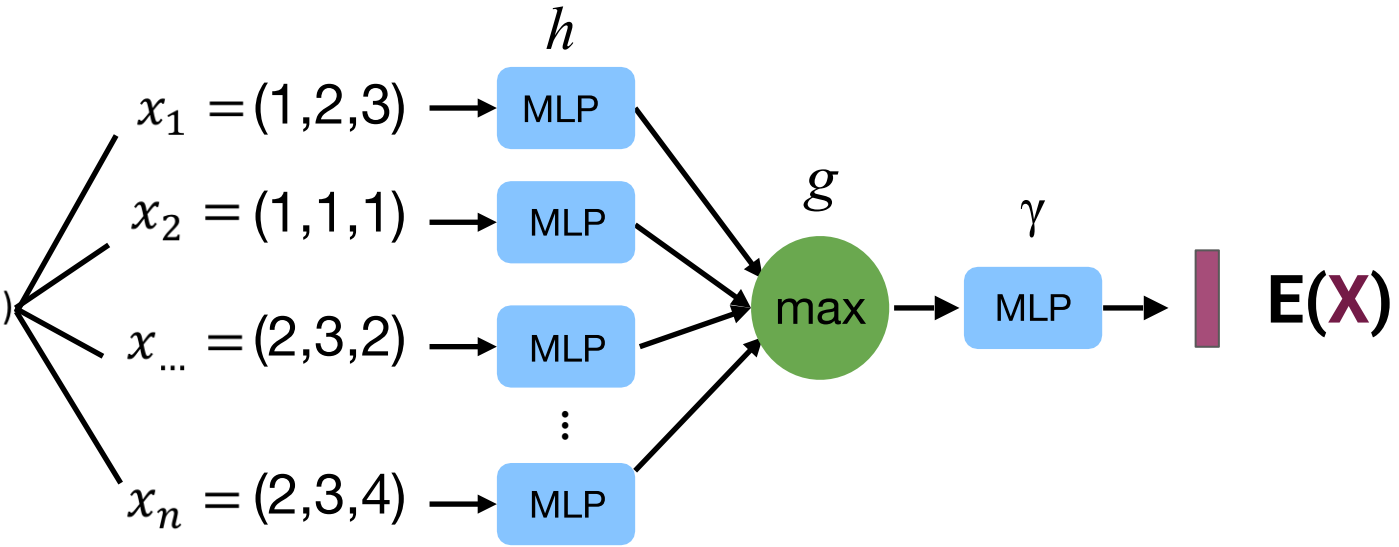


PointNet [Qi2017]



Input pointcloud

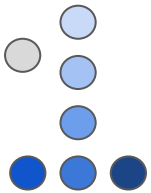
$\mathbf{X} = (x_1, x_2, \ldots, x_n)$

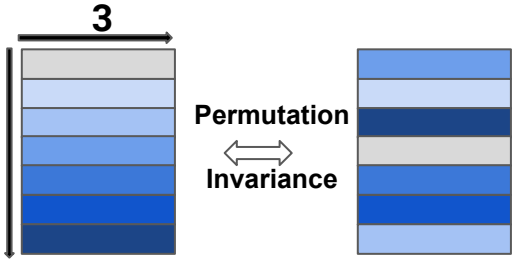


$$\mathbf{E}((x_1, x_2, \ldots, x_n)) = \gamma(\, g(\, h(\, x_1\,), \ldots, \, h(\, x_n\,)\,)\,)$$

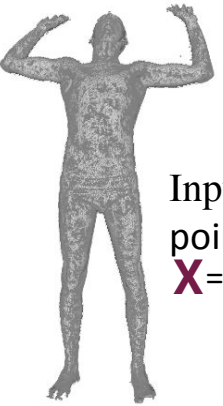
Encoder

E



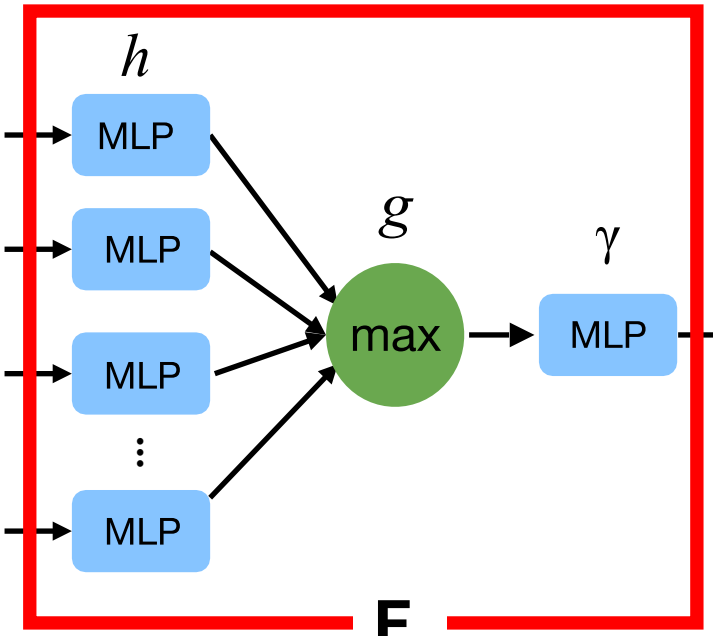


PointNet [Qi2017]



Input pointcloud

$\mathbf{X} = (x_1, x_2, \ldots, x_n)$

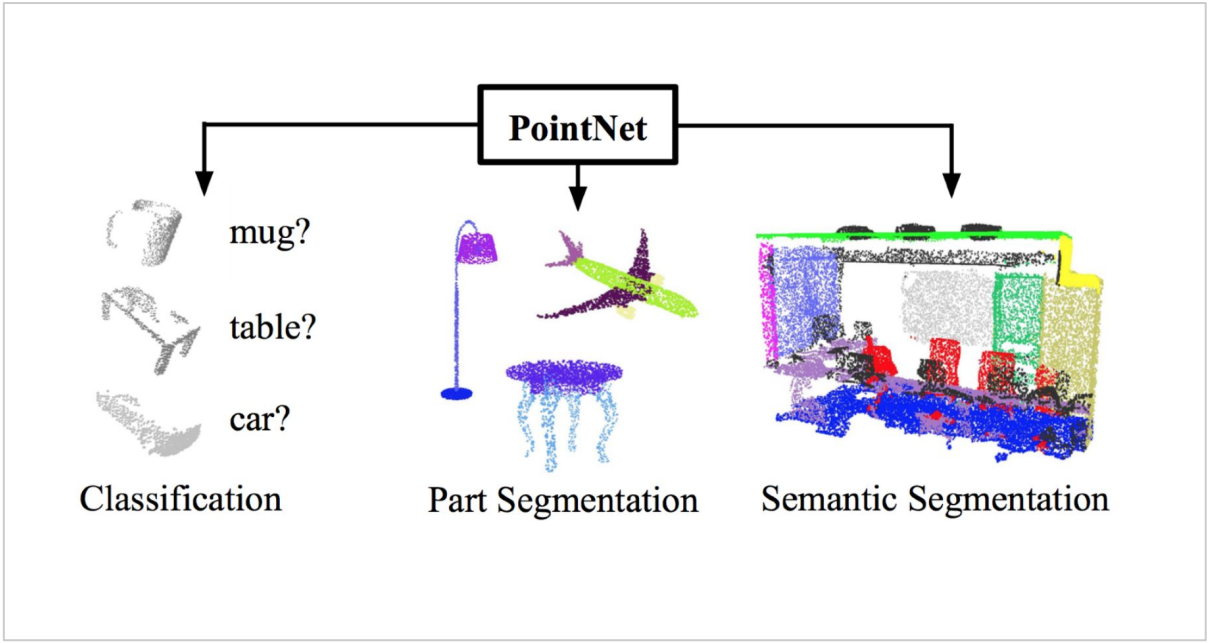


$$\mathbf{E}((x_1, x_2, \ldots, x_n)) = \gamma(\, g(\, h(\, x_1), \ldots, \; h(\, x_n)\,)\,)$$

Encoder

E

Results : Unified framework for various tasks

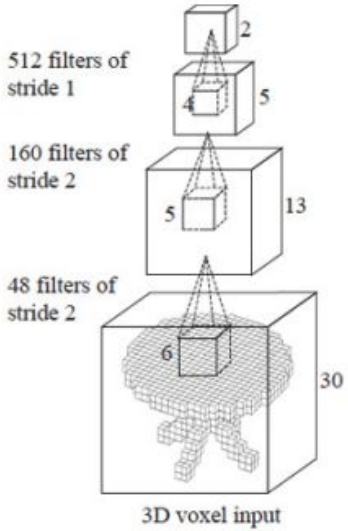


Credit [**Qi2017**]

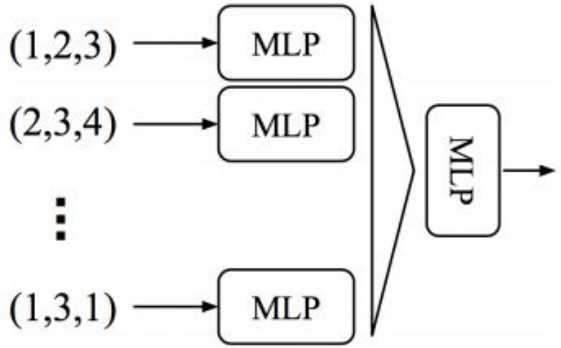
PointNet Limitations Credit [**Qi2017**]

- Hierarchical Feature Learning
- Increasing receptive field

Global Feature Learning
Receptive field:
one point OR all points



v.s.



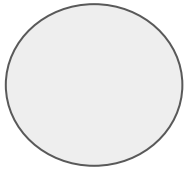
3D CNN (Wu et al.)

PointNet (vanilla) (Qi et al.)

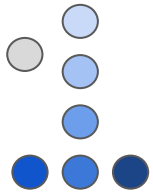
Encoder

E

Key idea : Global informations is computed in 1 stage : the max function.



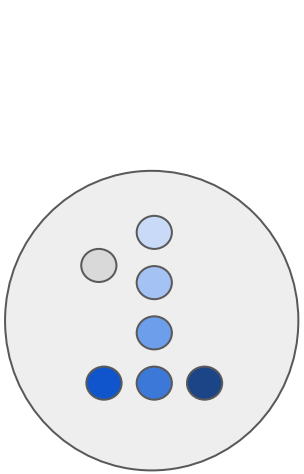
= PointNet Module

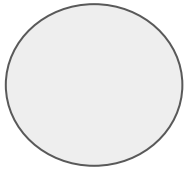


Encoder

E

Key idea : Global informations is computed in 1 stage : the max function.





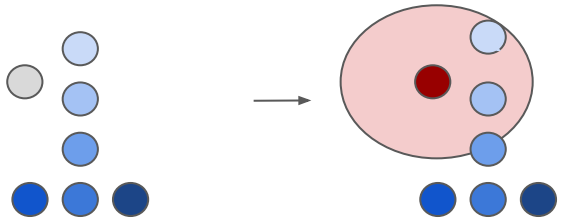
=

PointNet Module

E(X)

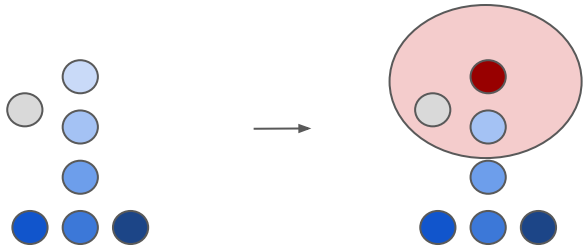
Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



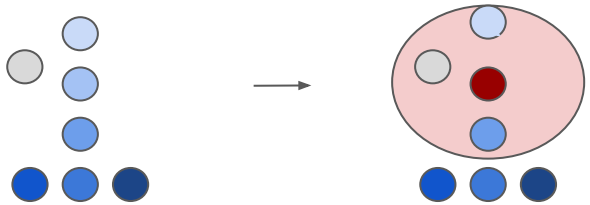
Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



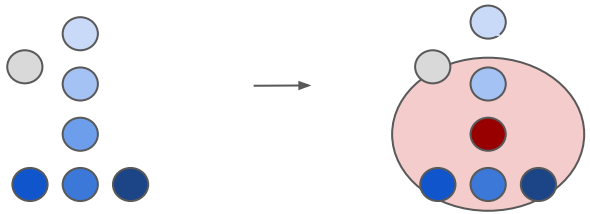
Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



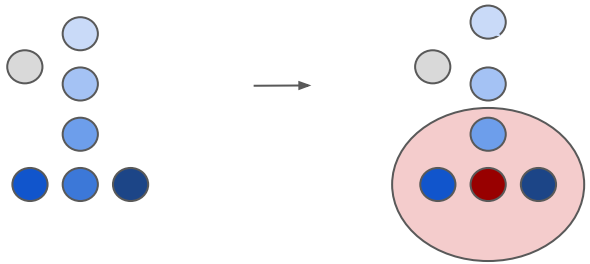
Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



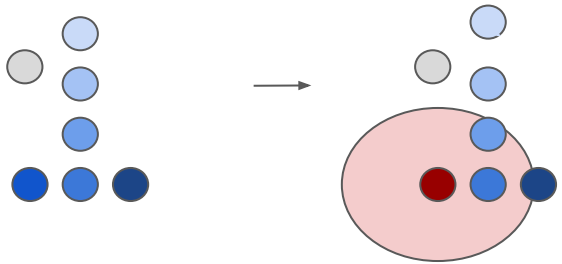
Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



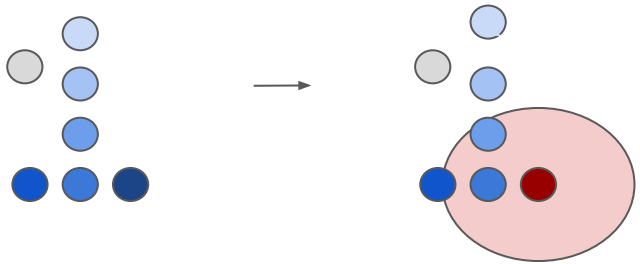
Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?

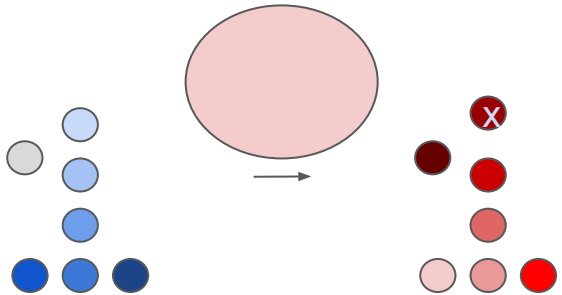


Encoder

E

Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?

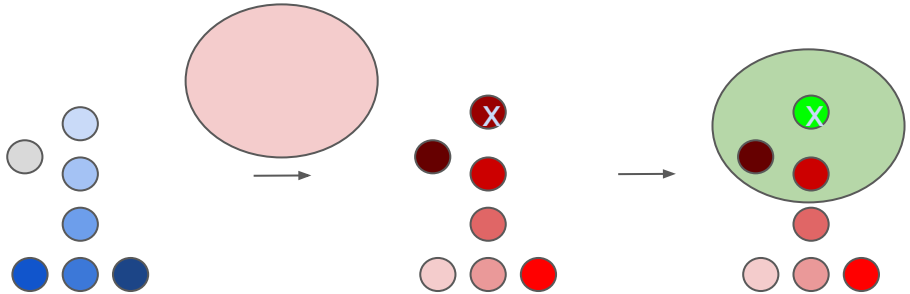


Encoder

E

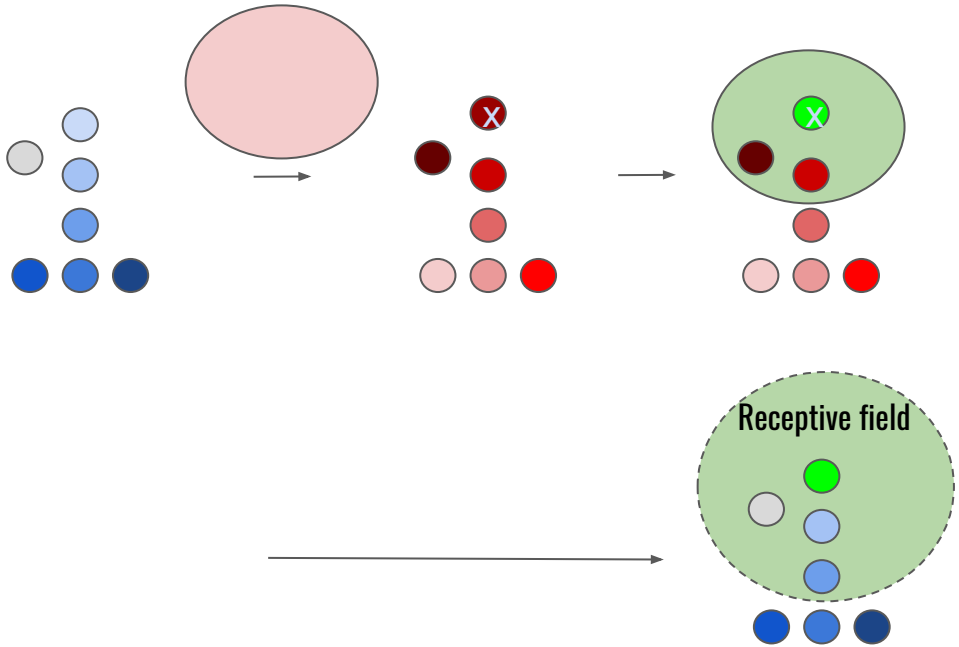
Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?

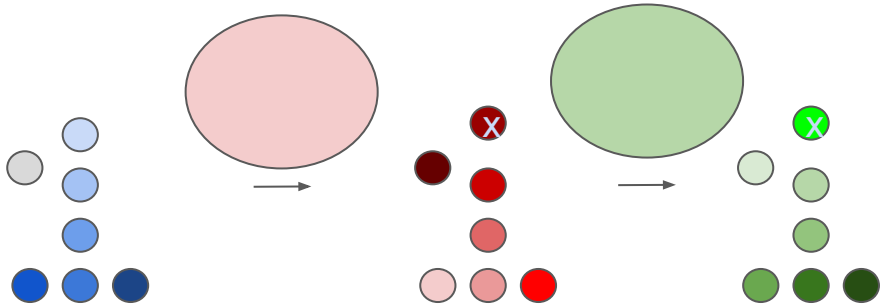


Encoder

E

Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?

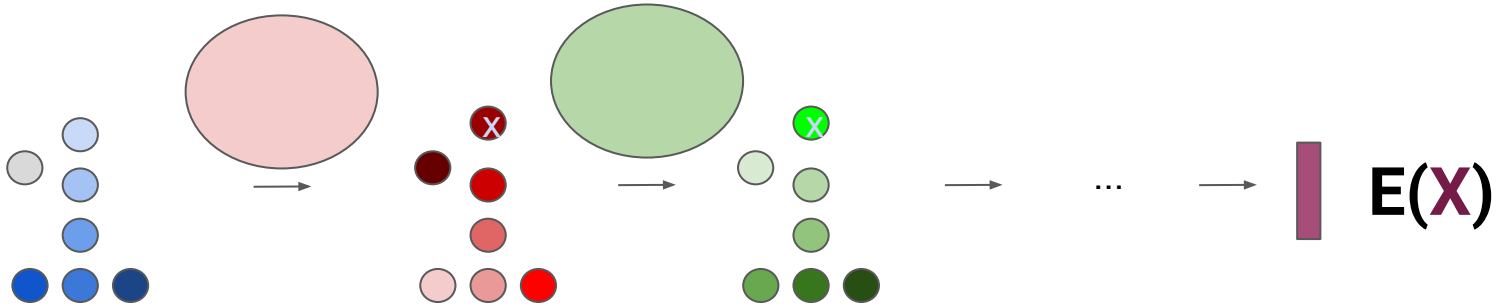


Encoder

E

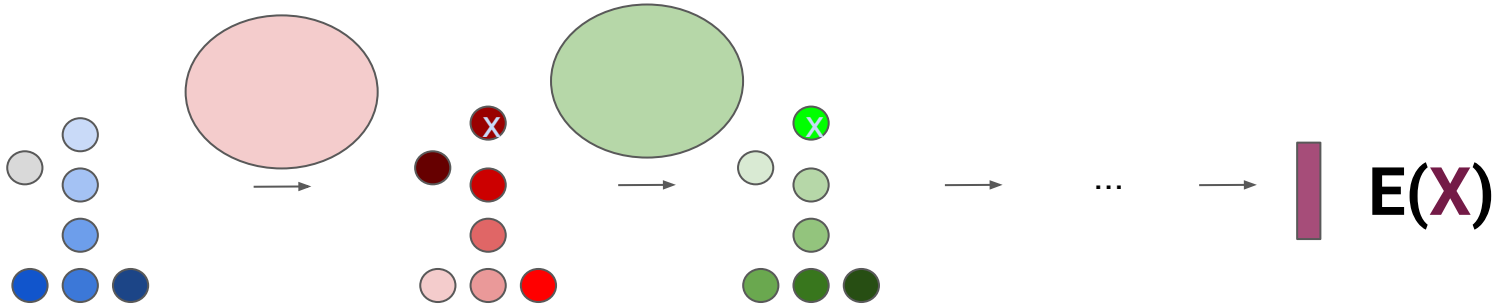
Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



Encoder
E

Key idea : Global informations is computed in 1 stage : the max function.
Inspired by their success in images, can we build hierarchical filters ?



Key considerations :

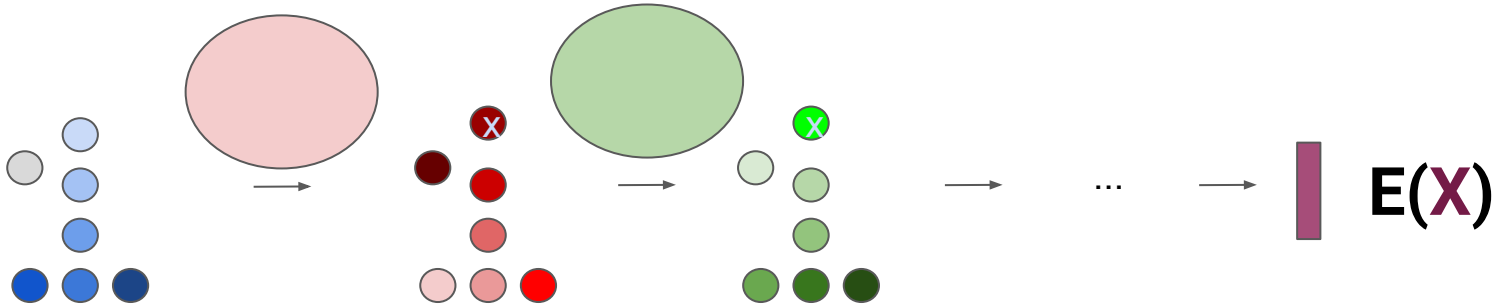
- Define a receptive field : Ball Query(PointNet++ [Qi2017b, Simonovsky2017]) ? Nearest Neighbors ? Nearest Neighbors in 8 quadrant (pointSIFT [Jiang2018]) ?
- Choose a metric : Euclidean ? Geodesic ?
- Choose the features : 3D input space features ? Current Layer features (Dynamic Graph CNN [Wang2018])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

Encoder

E

Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



Key considerations :

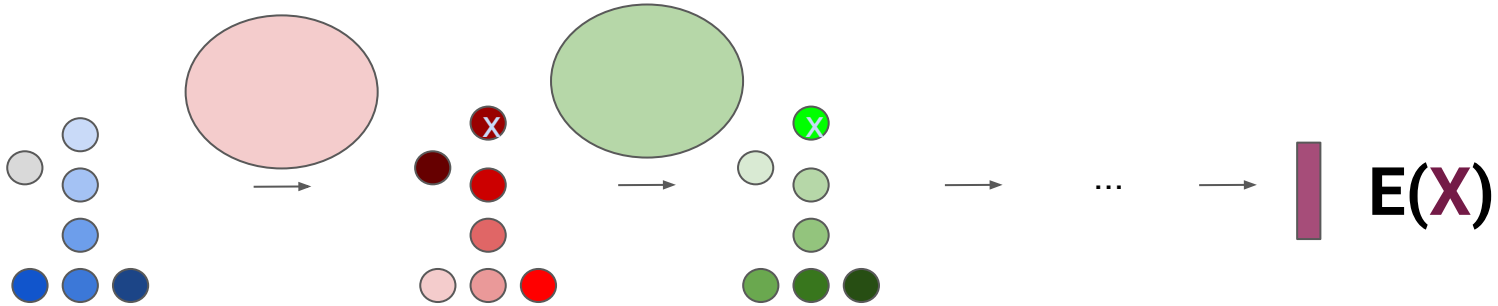
- Define a receptive field : Ball Query(PointNet++ [Qi2017b, Simonovsky2017]) ? Nearest Neighbors ? Nearest Neighbors in 8 quadrant (pointSIFT [Jiang2018]) ?
- Choose a metric : Euclidean ? Geodesic ?
- Choose the features : 3D input space features ? Current Layer features (Dynamic Graph CNN [Wang2018])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

Encoder

E

Key idea : Global informations is computed in 1 stage : the max function.

Inspired by their success in images, can we build hierarchical filters ?



Key considerations :

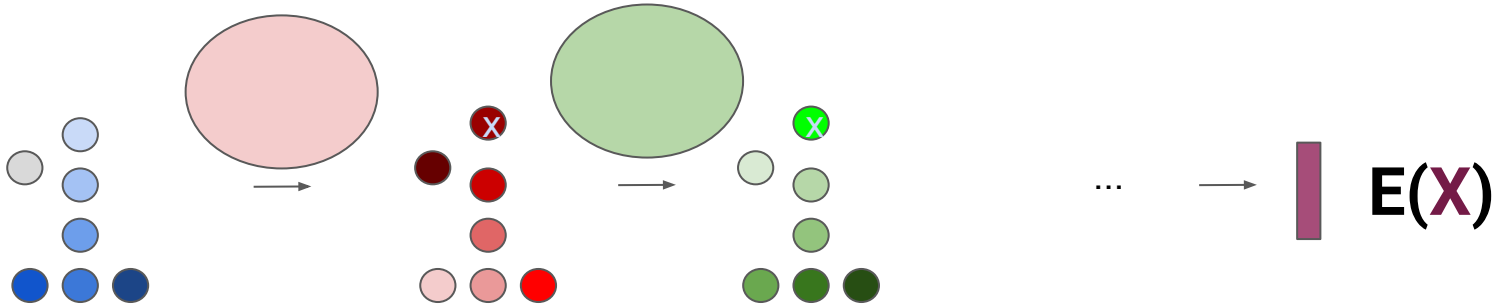
- Define a receptive field : Ball Query(PointNet++ [Qi2017b, Simonovsky2017]) ? Nearest Neighbors ? Nearest Neighbors in 8 quadrant (pointSIFT [Jiang2018]) ?
- Choose a metric : Euclidean ? Geodesic ?
- Choose the features : 3D input space features ? Current Layer features (Dynamic Graph CNN [**Wang2018**]) ?
- Global coordinates ? Local coordinates [Qi2017b, Wang2018]?

Encoder

E

Key idea : Global informations 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]?

Encoder

E

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)

...

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)

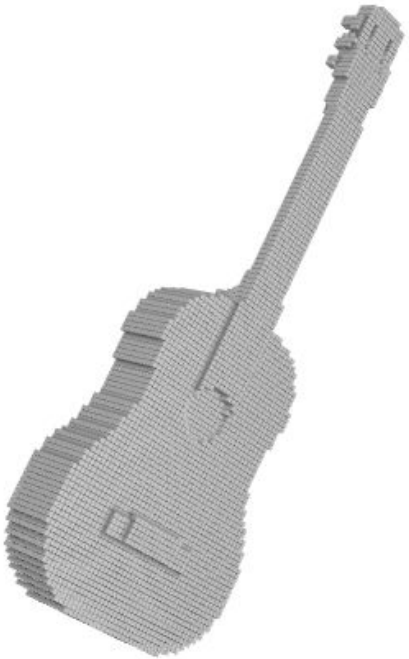
3D Object **Y**

Encoder
E

Voxels 3d-r2n2 [Choy2016], Voxnet [Maturana2015], [Qi2016], [Wu2015]

Decoder
D

- ➔ 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

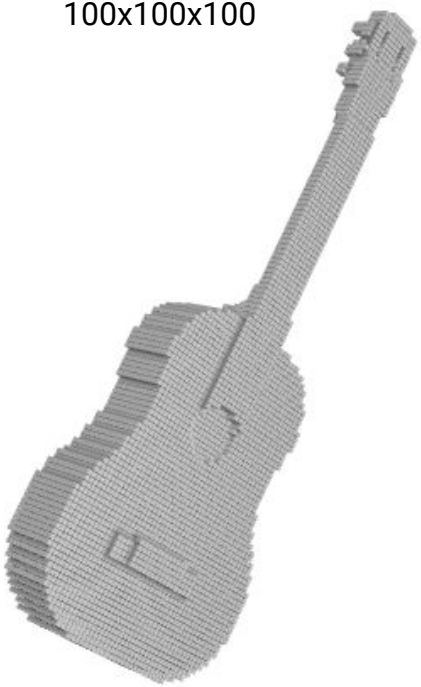


Volumetric representations

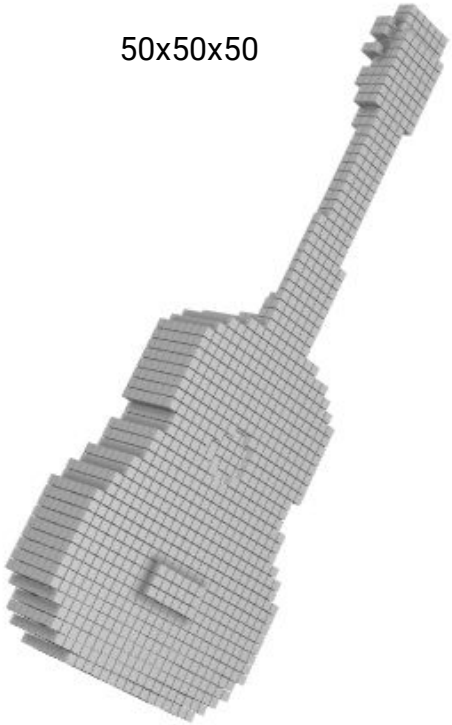
Encoder
E

Decoder
D

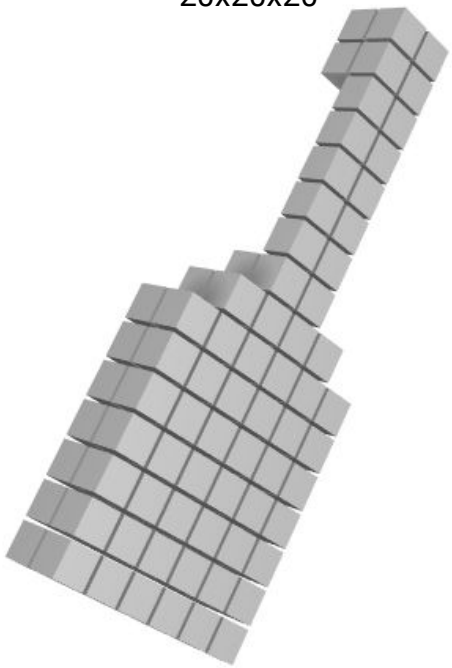
100x100x100



50x50x50



20x20x20



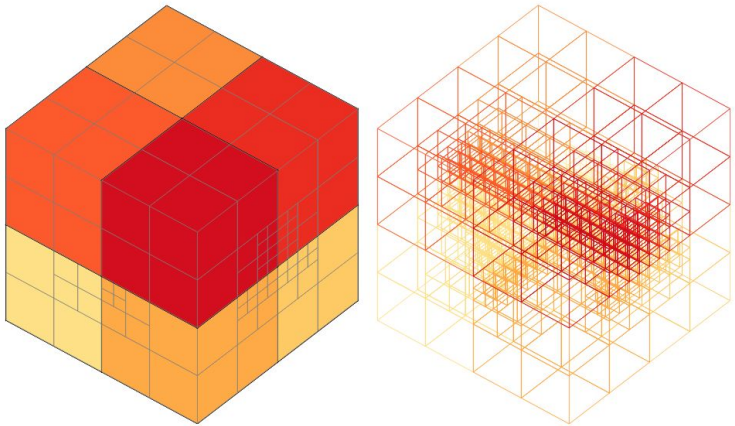
Encoder
E

Hybrid Grid-Octree Data Structure

Decoder
D

Octnet **[Riegler2017]**, OGN **[Tatarchenko2017]**

- ➔ Grid of octrees with fixed small depth : typically 3
- ➔ Computationally more effective
- ➔ Good compression rate



OctNet input

Encoder
E

Decoder
D

- ➔ 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

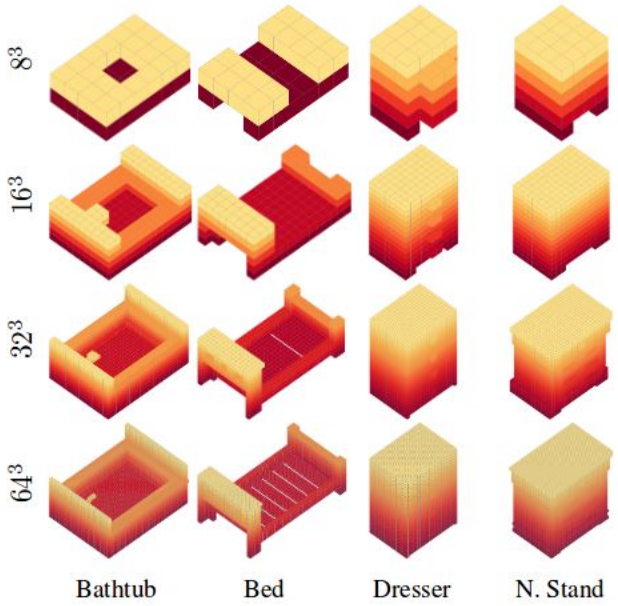


Figure 8: **Voxelized 3D Shapes from ModelNet10.**

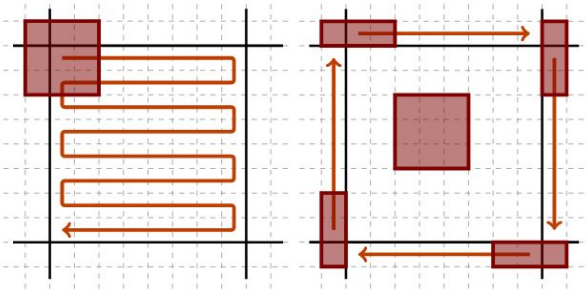
Convolutions on Grid-Octree Data Structure

Encoder E

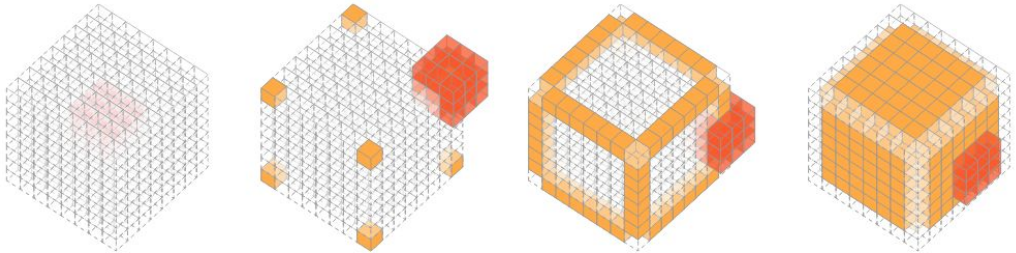
Decoder D

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

➔ Convolution is computed on the boundaries



(a) Standard Convolution (b) Efficient Convolution



(a) Constant (b) Corners (c) Edges (d) Faces

Figure 14: **Efficient Convolution.**

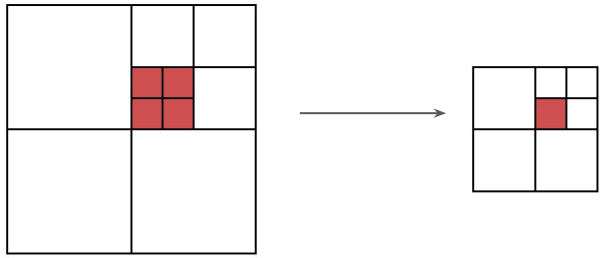
Encoder

E

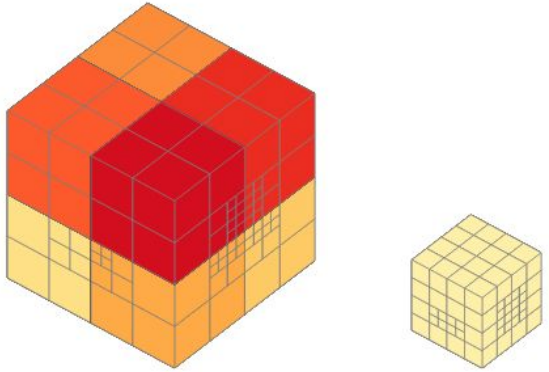
Pooling on Grid-Octree Data Structure

- ➔ Voxels at maximum resolution are pulled
- ➔ Voxels at higher resolutions are halved in size

2D example



3D



(a) Input (b) Output

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)

Decoding towards an octree

Decoder
D

Objective : Predicting the occupancy value of each cell in the octree

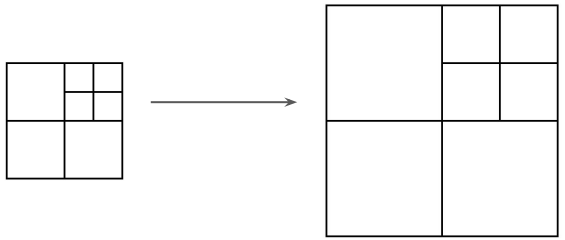
Issue : Contrarily to voxels, the octree structure is specific to each sample

➔ We need to predict the octree structure

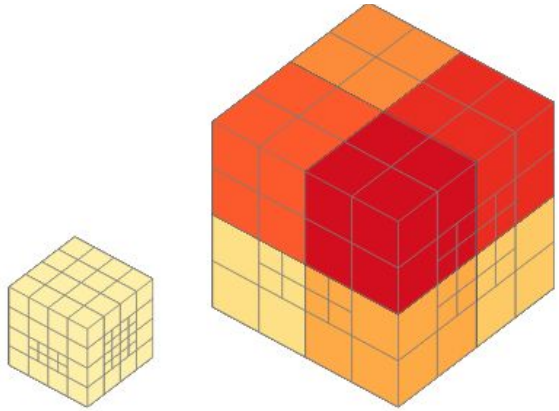
Unpooling on Grid-Octree Data Structure

➔ All nodes double their sizes

2D example



3D



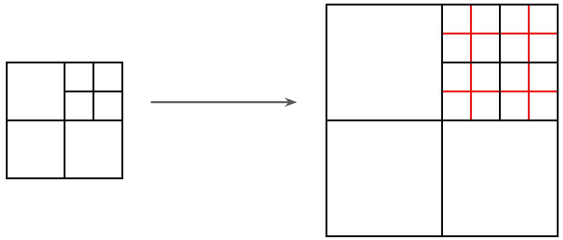
(a) Input (b) Output

Unpooling on Grid-Octree Data Structure

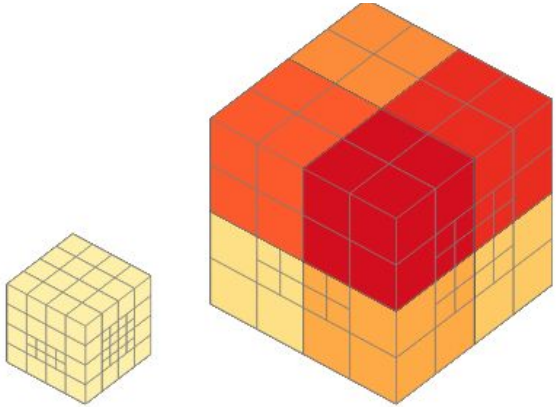
➔ All nodes double their sizes

What about capturing details at finer resolution ?

2D example



3D



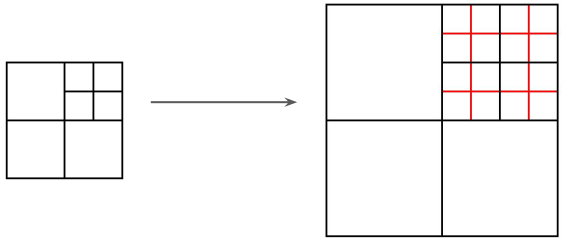
(a) Input (b) Output

Unpooling on Grid-Octree Data Structure

Decoder
D

➔ All nodes double their sizes

2D example

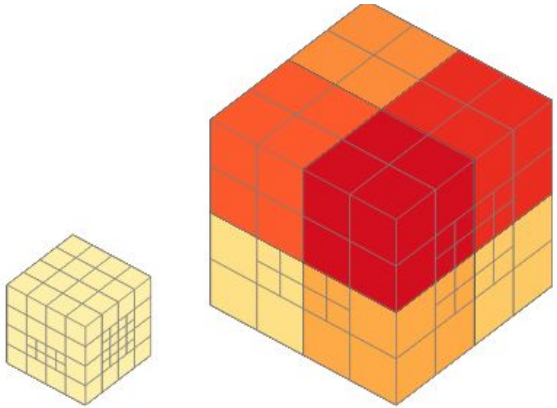


What about capturing details at finer resolution ?

➔ 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 splitted in 8 to capture finer details **[Tatarchenko2017]**

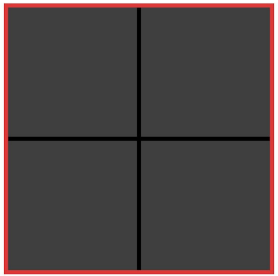
3D



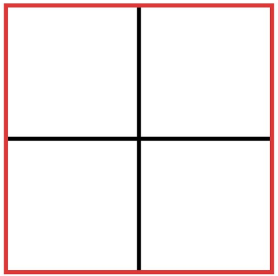
Octree generating networks - results

Decoder
D

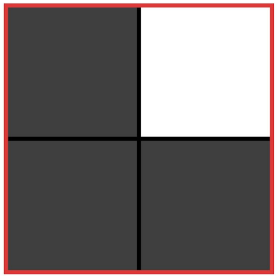
Subdivision is predicted as a classification task. **[Tatarchenko2017]**



Full



Empty



Mixed
(any other configuration)

➔ This can be supervised **at each layer** of the network because we know whether a subdivision occurs or not in the ground truth.

The red cell can either be

- full or empty: we don't subdivide
- mixed: we subdivide

Octree generating networks - results

Decoder
D

OGN **[Tatarchenko2017]**

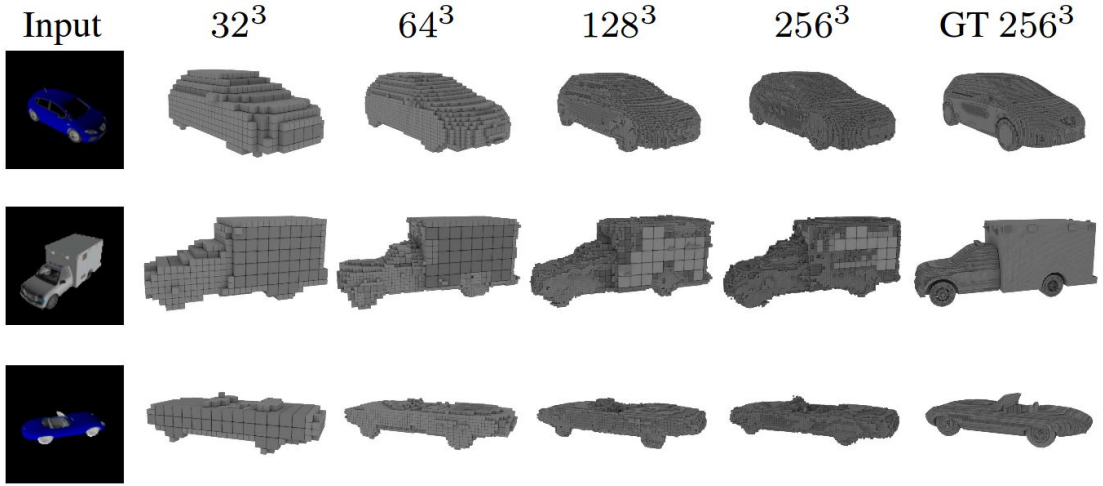


Figure 8. Single-image 3D reconstruction on the ShapeNet-cars dataset using OGN in different resolutions.

Octree-based reconstruction

Encoder
E

Decoder
D

- ➔ 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**

Encoder E

Feature vector

Decoder D

Choice of representation

Volumetric (OctNet)

PointClouds (PointSetGen)

Surfaces (AtlasNet)

Signed Distance Function

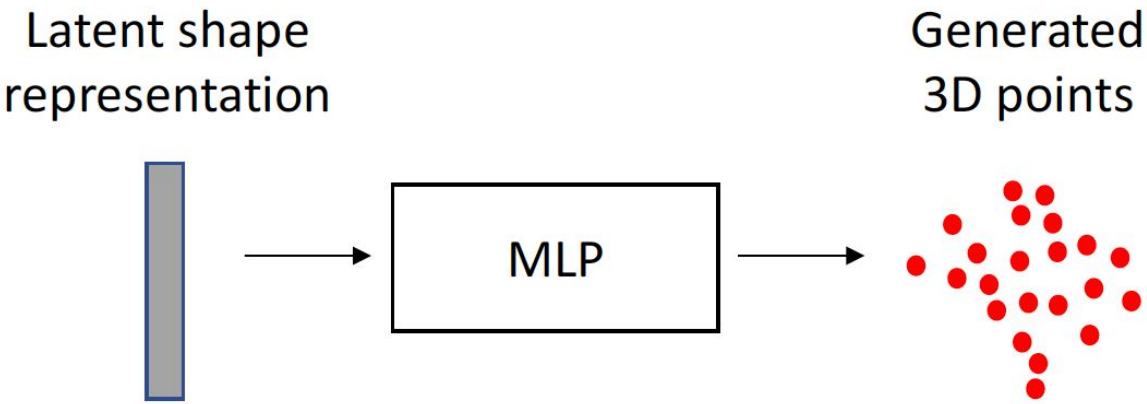
Geometric Primitives

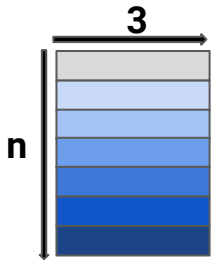
Loss

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

Generating points PointSetGen[**Fan2017**]

Decoder
D





Training setup for 3D reconstruction

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

...

3D Object **Y**

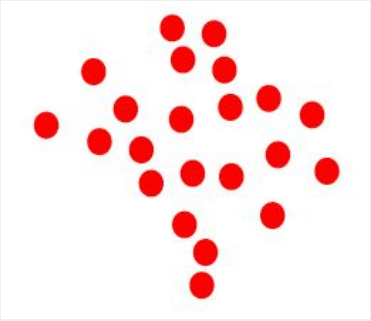
Encoder
E

Feature vector

Decoder
D

Choice of representation

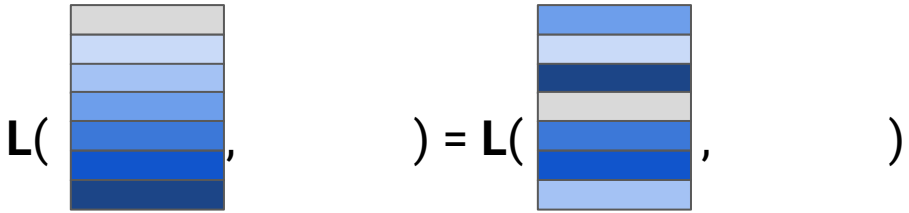
PointClouds (PointSetGen)



Loss ?

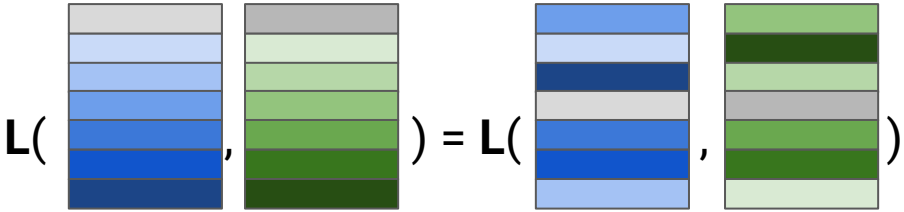
L(D(E(X)),Y)

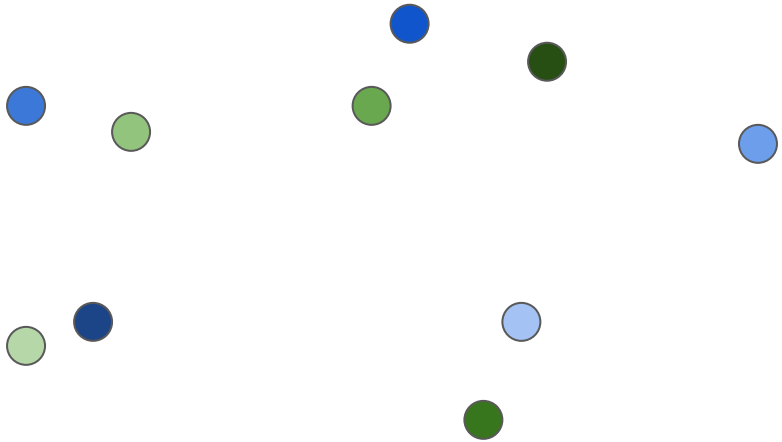
Loss on pointclouds



	Complexity
EMD	n^3
Chamfer	n^2

Loss on pointclouds



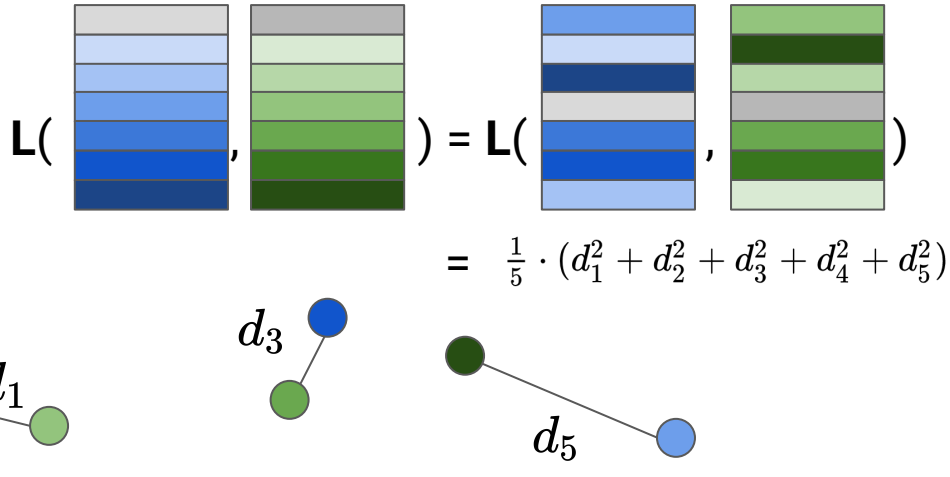


	Complexity
EMD	n^3
Chamfer	n^2

Loss on pointclouds

Find the **optimal assignement** and compute **Earth Mover Distance** (**EMD**)

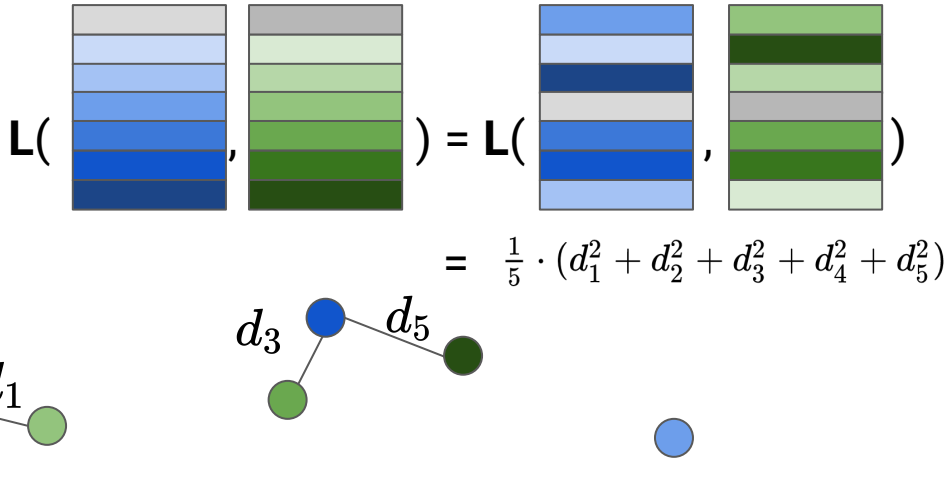
- ➔ Hungarian Algorithm [**Kuhn1955**] ~O(n³)
- ➔ Simplex based solver through LP formulation ~O(Hungarian)
- ➔ Sinkhorn regularization [**Cuturi2013**] in near linear time [**Altschuler2017**]
- ➔ (1+ε) approximation [**Bertsekas1988**] in ~O(n³)



	Complexity
EMD	**n³**
Chamfer	n²

Loss on pointclouds

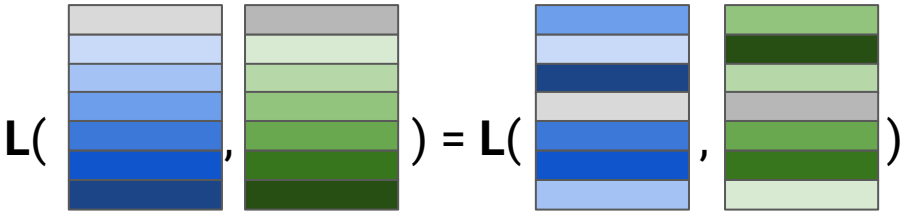
Find the **nearest neighbours** and compute **Chamfer Distance (CD) = L(,) +**



	Complexity
EMD	**n^3**
Chamfer	**n^2**

Loss on pointclouds

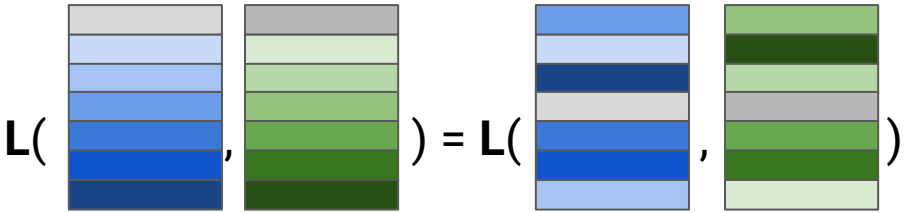
Find the **nearest neighbours** and compute
Chamfer Distance (CD) = L(,) +



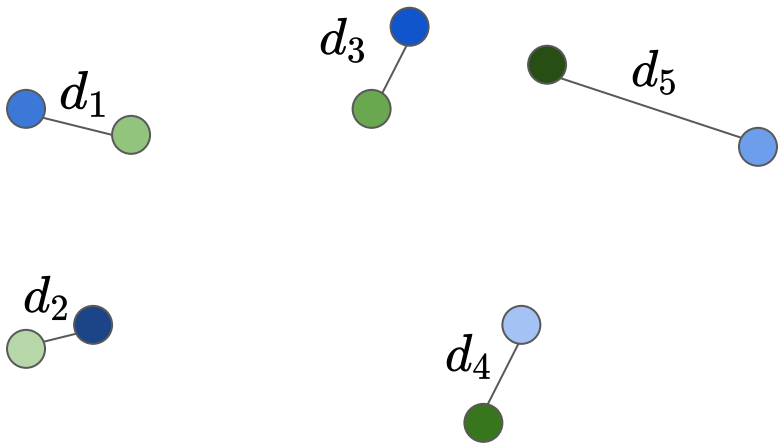
	Complexity
EMD	**n^3**
Chamfer	**n^2**

Loss on pointclouds

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



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

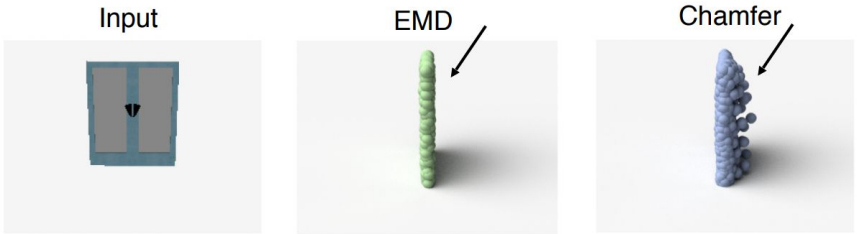


	Complexity
EMD	**n^3**
Chamfer	**n^2**

Loss on pointclouds : the mean shape carries characteristics of the distance metric

Distribution $\mathbb{S}$ of pointclouds of varying radius

$$\bar{x} = \underset{x}{\operatorname{argmin}}\ \mathbb{E}_{s\sim\mathbb{S}}[d(x,s)]$$



Credit : [**Fan2016**]

Training setup for 3D reconstruction

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

...

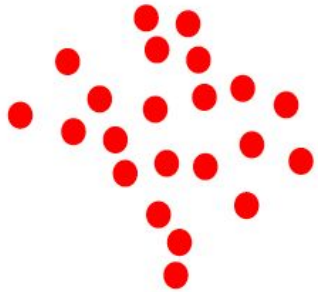
Encoder E

Feature vector

Decoder D

Choice of representation

PointClouds (PointSetGen)



Loss : Chamfer Distance

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

3D Object **Y**

Generating points

Encoder

E

Decoder

D

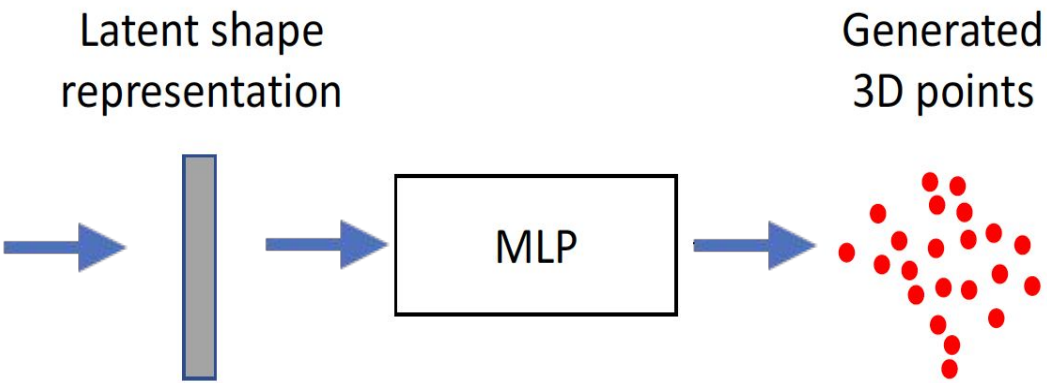
Test Shape

Generating points

Encoder
E

Decoder
D

Test Shape



Generating points

Encoder

E

Test Shape

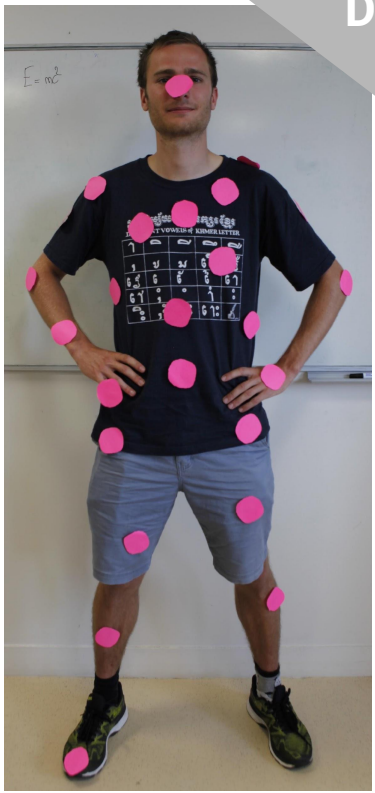
Latent shape representation

MLP

Generated 3D points

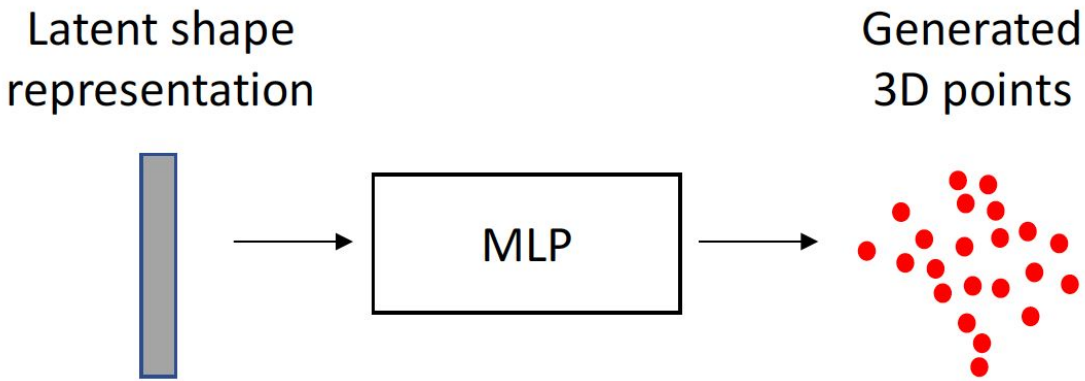
Decoder

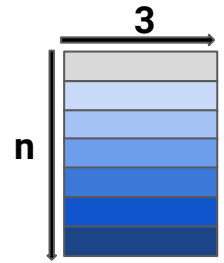
D



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



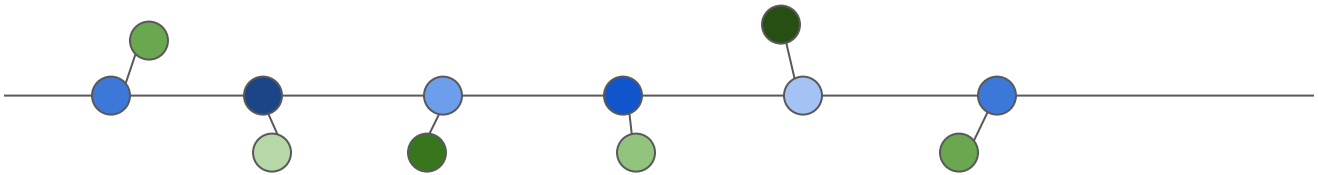


Limitation of PointSetGen [Fan2017]

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

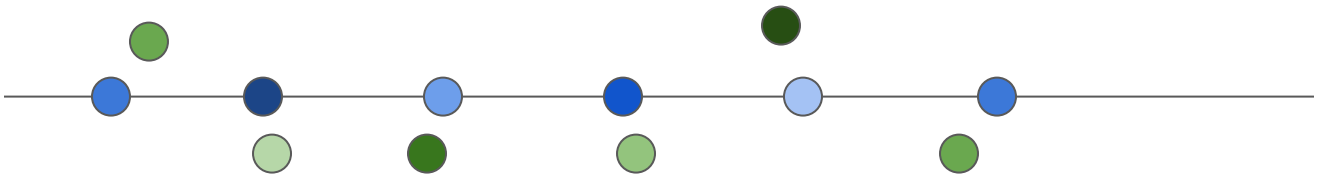
Limitation of PointSetGen [Fan2017]

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



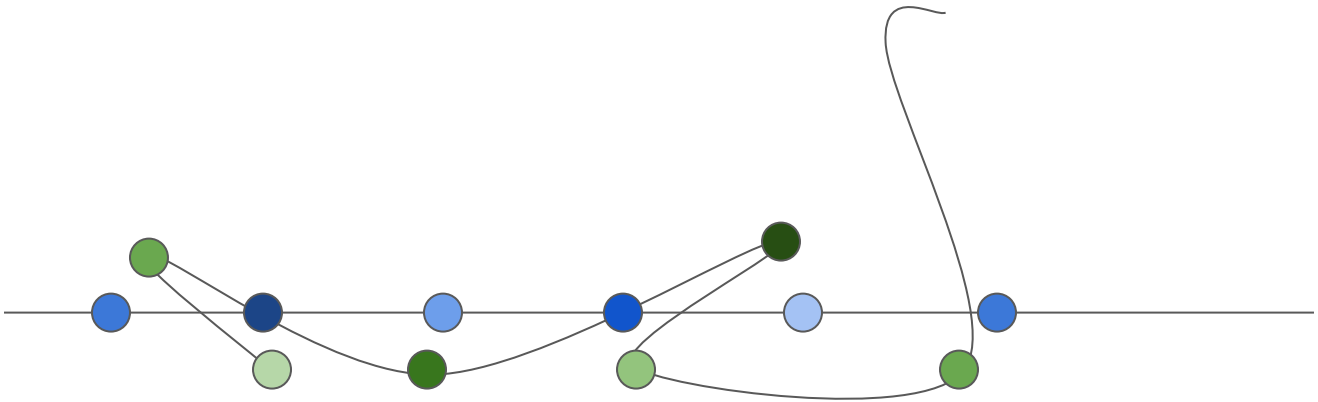
Limitation of PointSetGen [Fan2017]

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



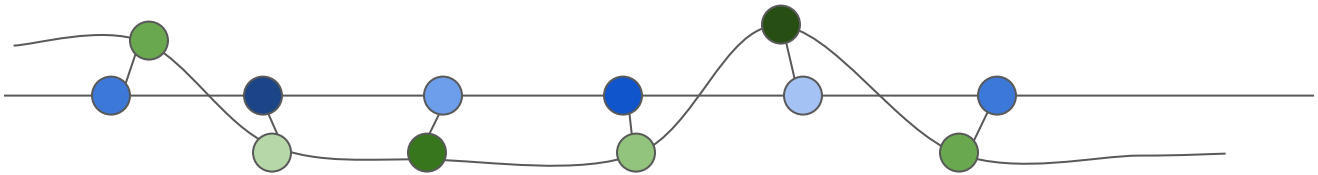
Limitation of PointSetGen [Fan2017]

- ➔ 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**

Encoder E

Feature vector

Decoder D

Choice of representation

Volumetric (OctNet)

PointClouds (PointSetGen)

Surfaces (AtlasNet)

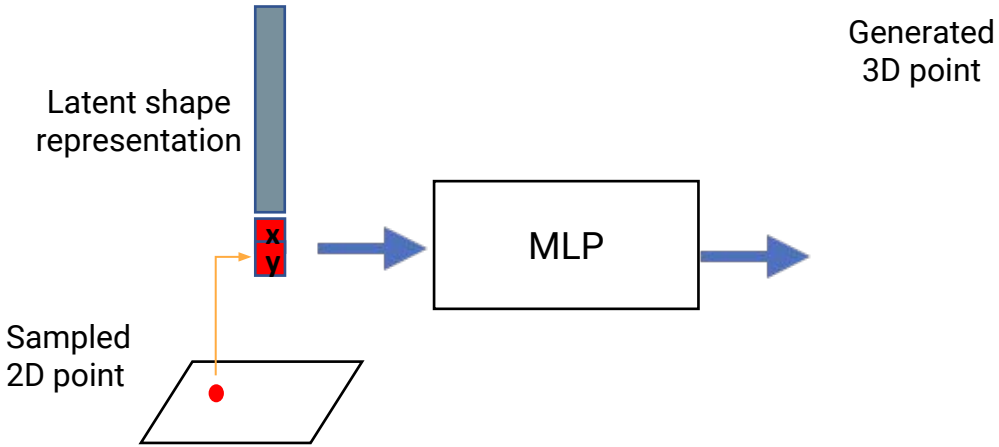
Signed Distance Function

Geometric Primitives

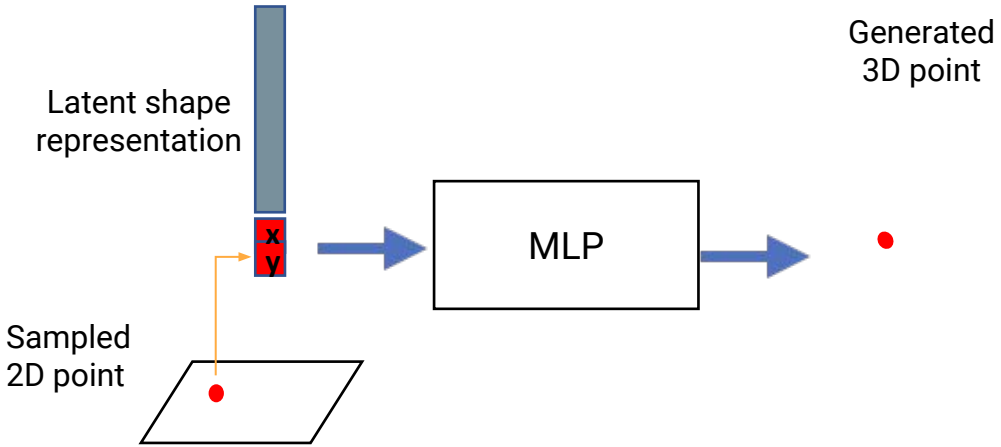
Loss

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

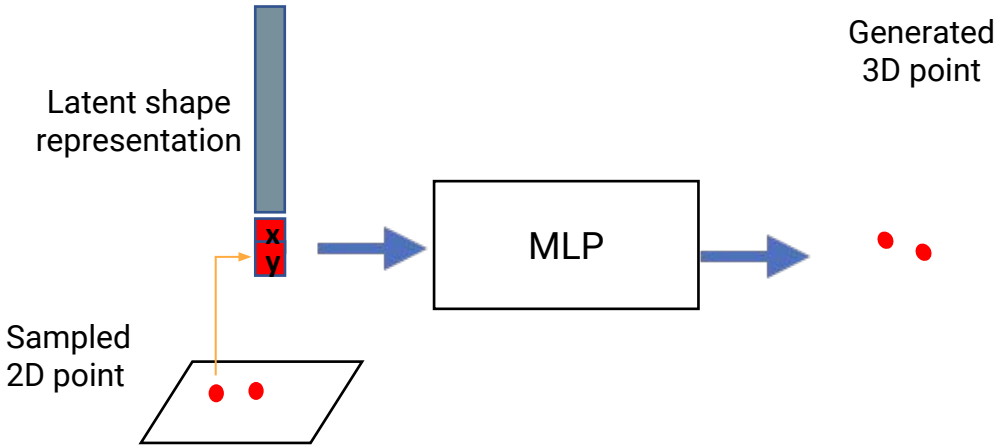
Deform a surface [Groueix2018]



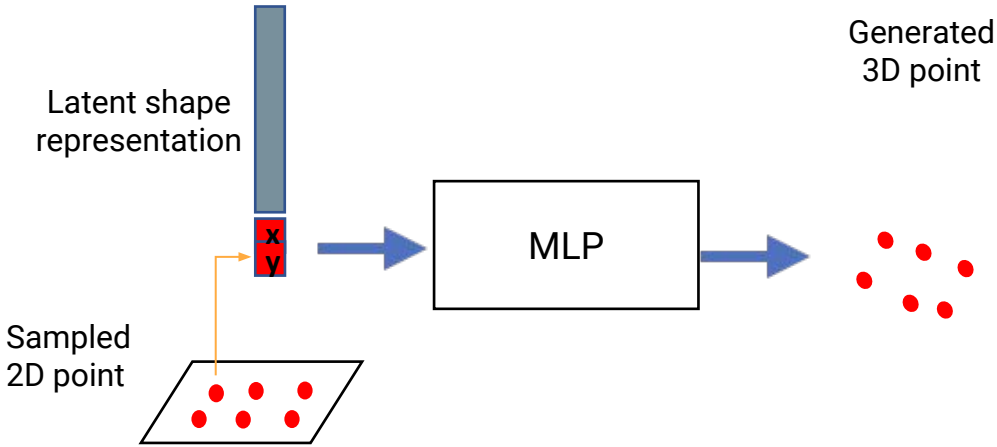
Deform a surface : space mapping trick [Groueix2018]



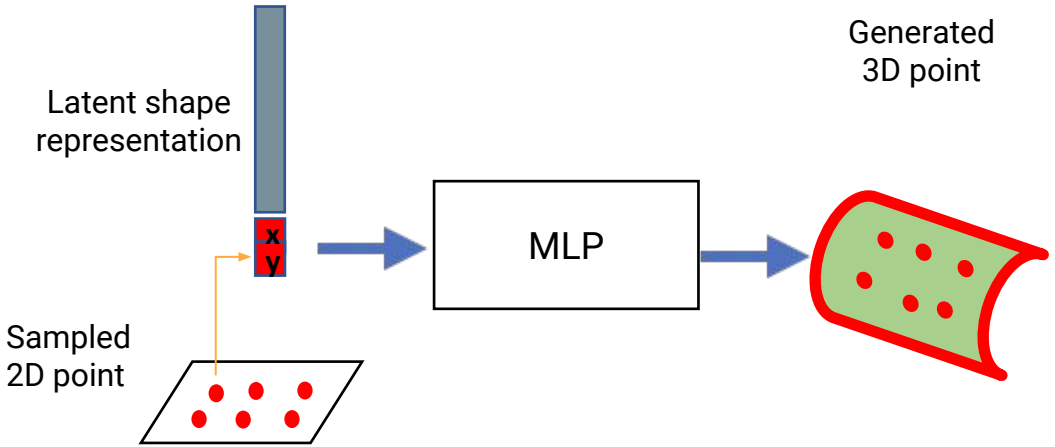
Deform a surface [Groueix2018]



Deform a surface [Groueix2018]

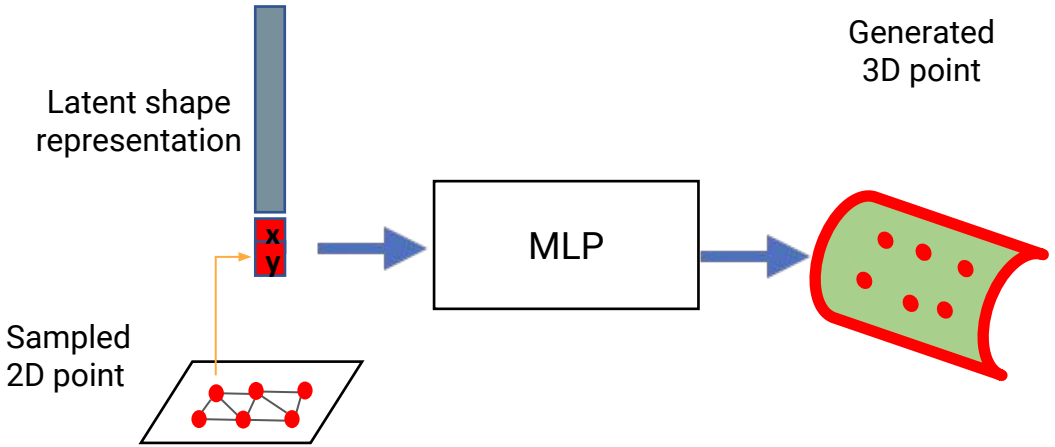


Deform a surface [Groueix2018]

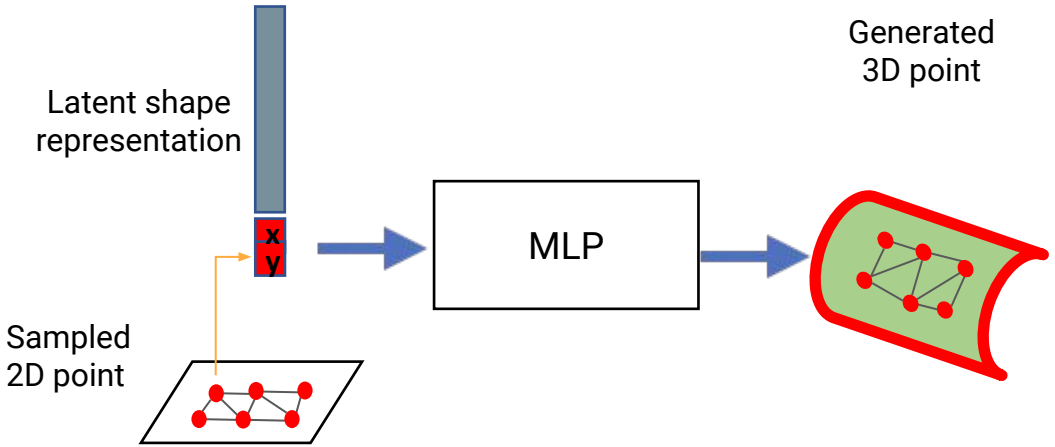


Deform a surface [Groueix2018]

Decoder
D



Deform a surface [Groueix2018]

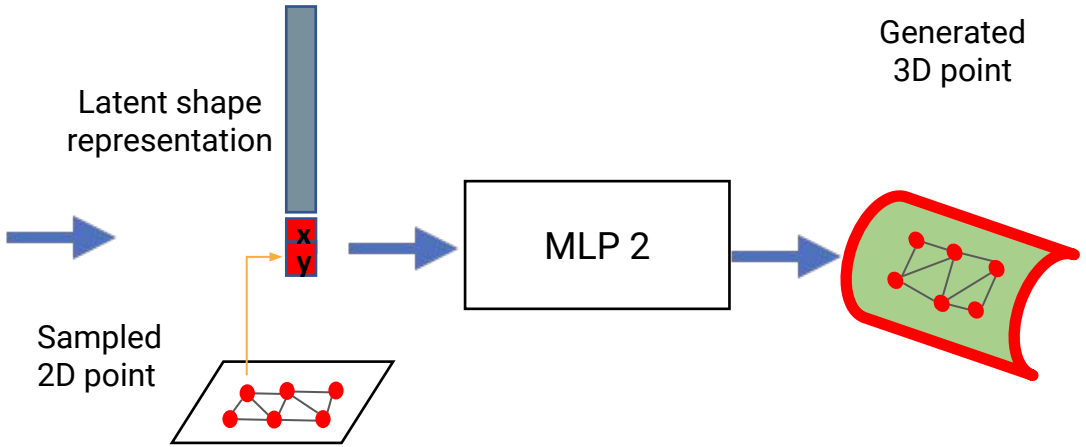


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape

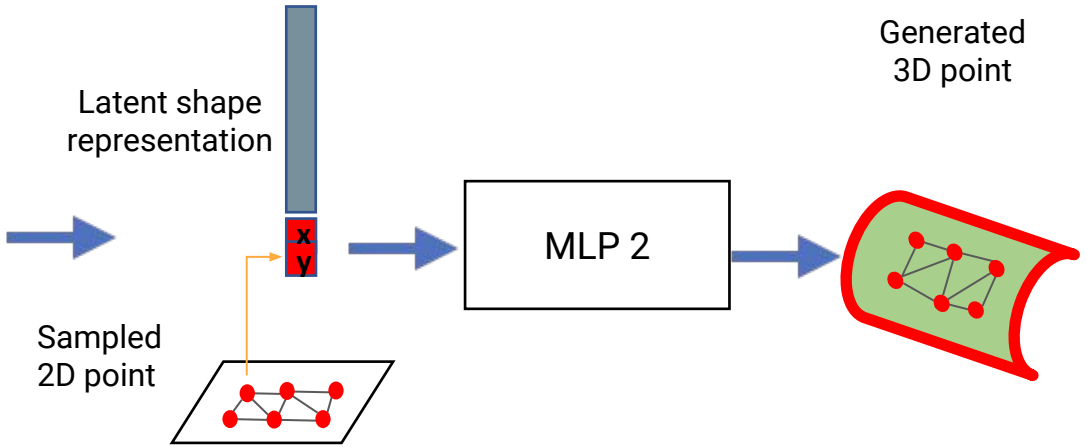


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape

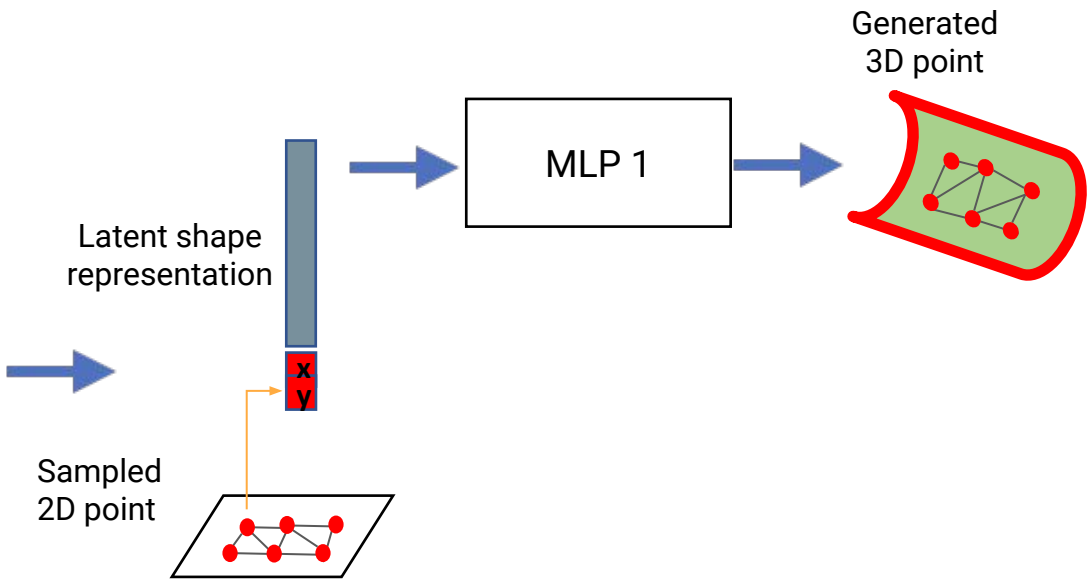


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape

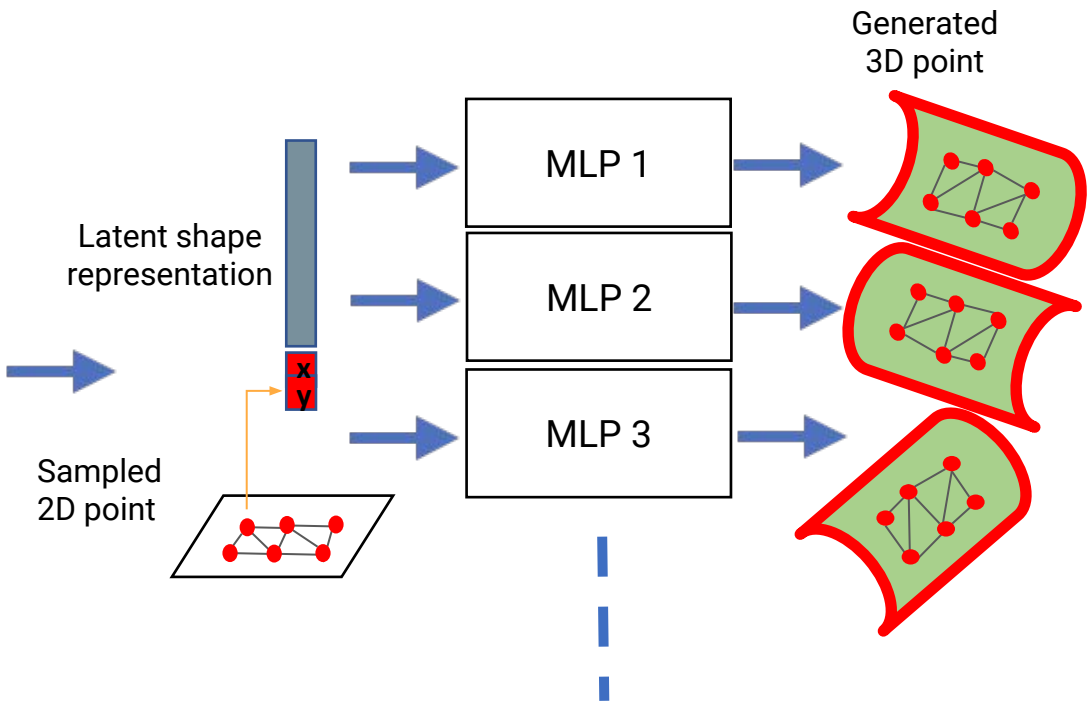


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape

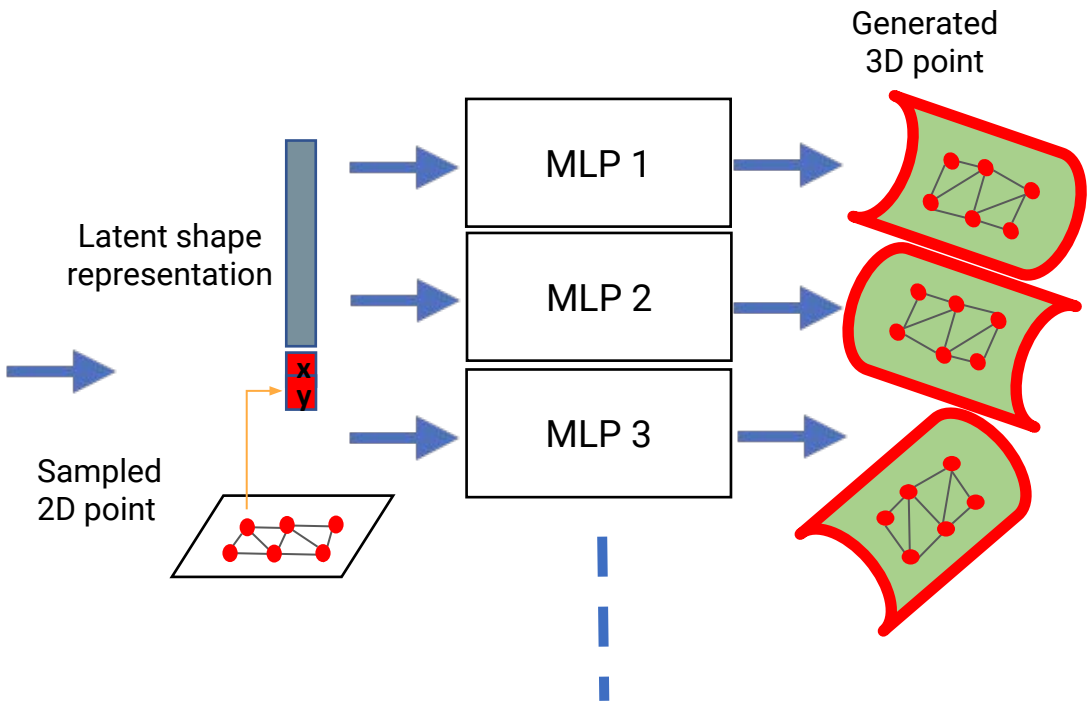


Deform a surface [Groueix2018]

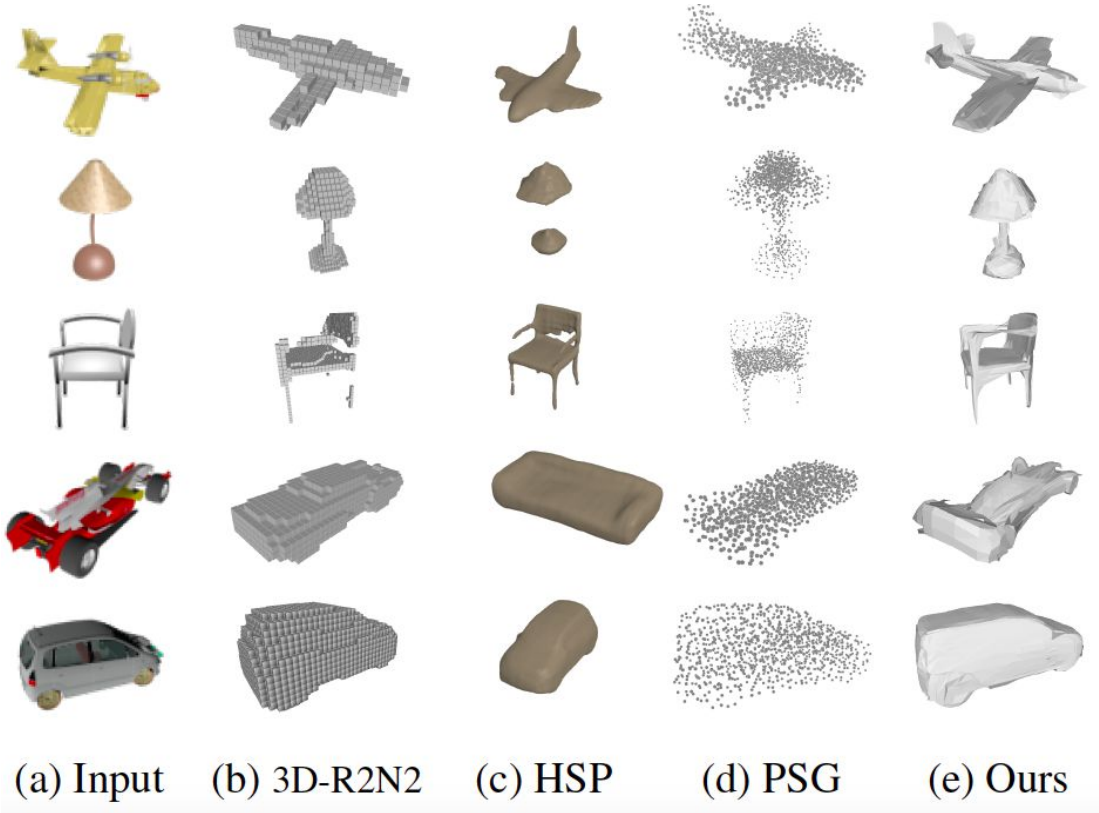
Encoder
E

Decoder
D

Test Shape



Results : Single View Reconstruction



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

Feature vector

Decoder
D

Choice of representation

Volumetric (OctNet)

PointClouds (PointSetGen)

Surfaces (AtlasNet)

Signed Distance Function

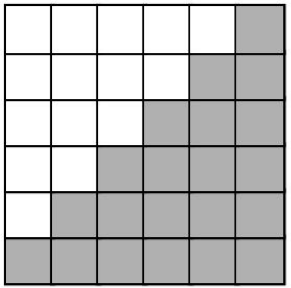
Geometric Primitives

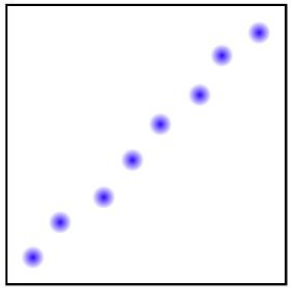
Loss

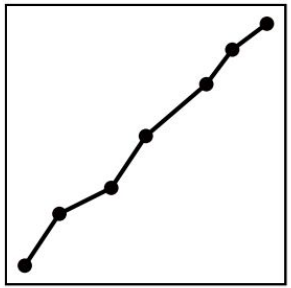
L(D(E(X)),Y)

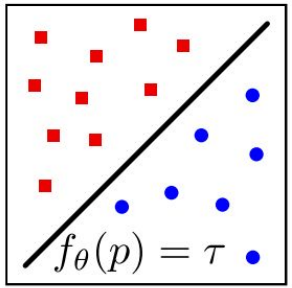
Can the space mapping trick be applied to volumetric representations ?

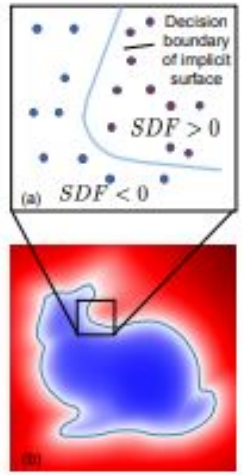
-> yes, through the Signed Distance Function (SDF) ! [Mescheder2018], [Park2019], [Chen2019]

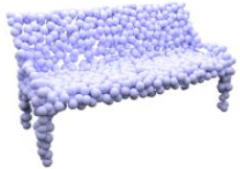








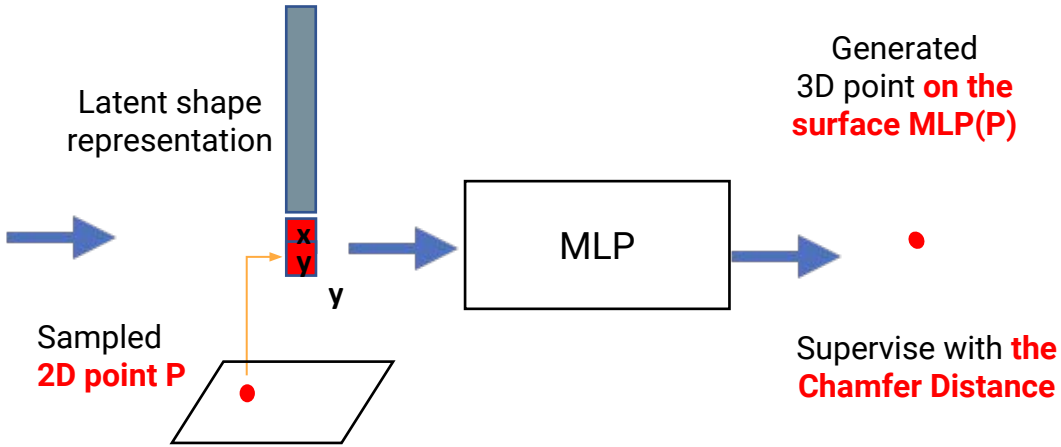




Credit:[Mescheder2018] a) Voxels b) Points c) Meshes d) Signed Distance Function

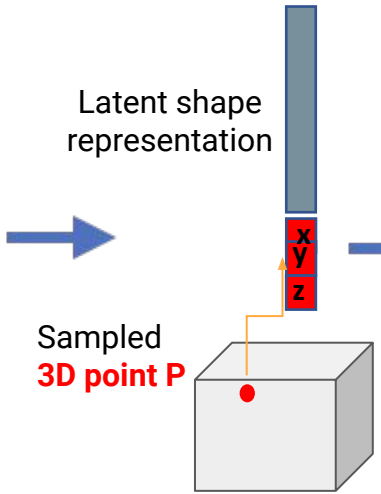
Deform a surface : space mapping trick [Groueix2018]

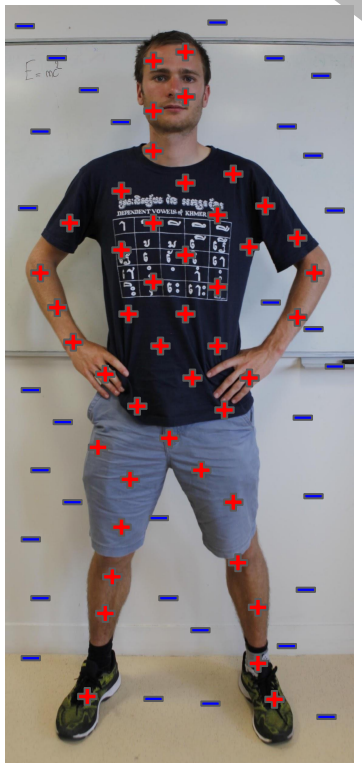
Decoder
D



Deform a volume [Mescheder2018]

Decoder
D





Deform a volume [Mescheder2018]

Decoder
D

Latent shape representation

x
y
z

Sampled
3D point P

MLP

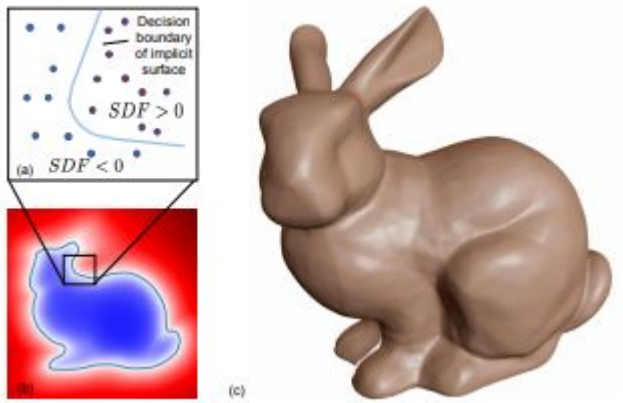
Predict Signed Distance Function : SDF(P)

SDF(P)

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

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 ?

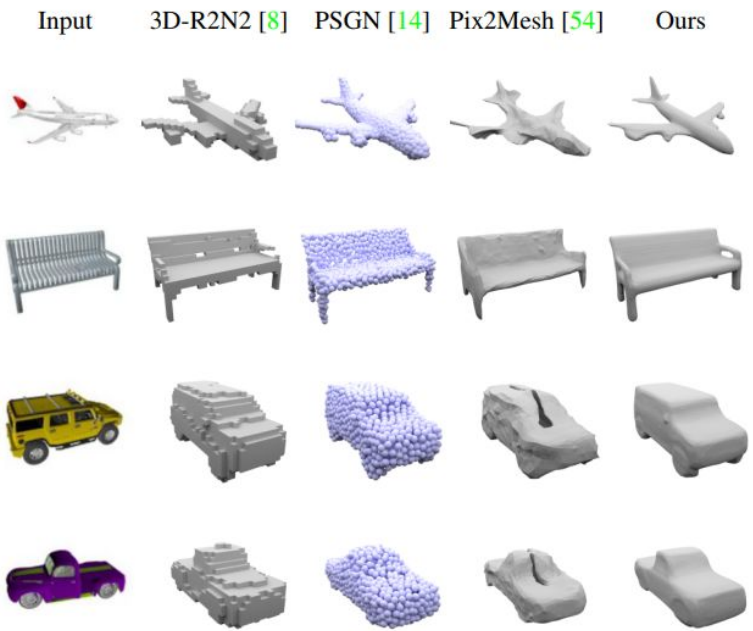
-> yes, through the Signed Distance Function (SDF) ! [Mescheder2018], [Park2019], [Chen2019]

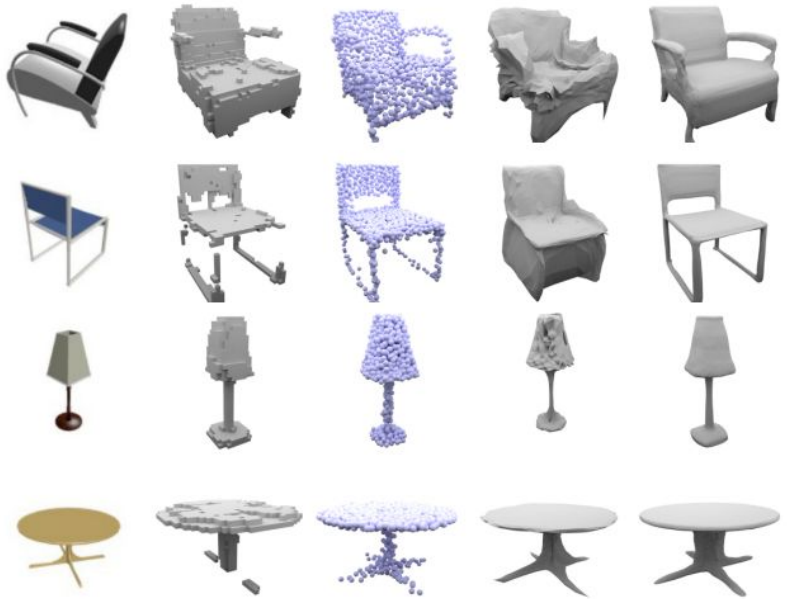
++ Get a voxel based representation at infinite granularity

++ Get analytic normals : dSDF(x)/dx

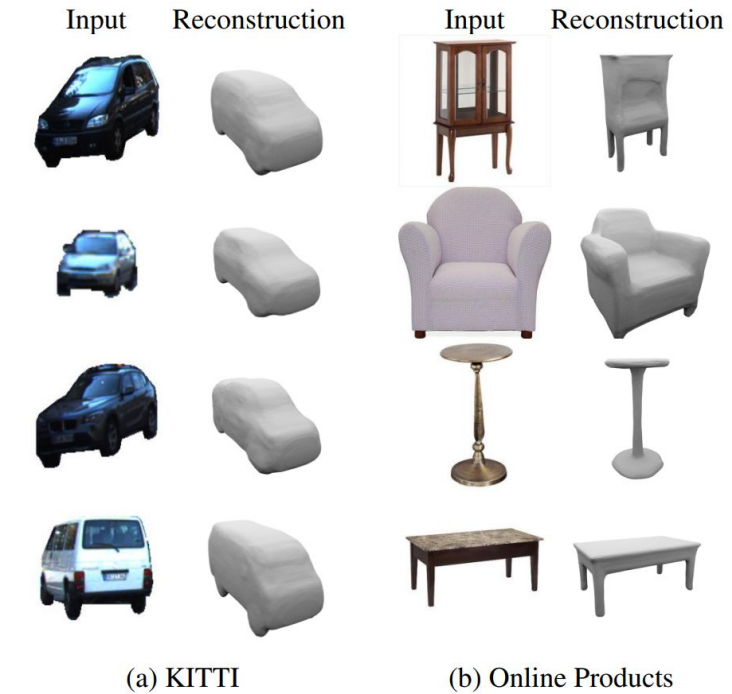
++ Topology is no longer an issue

Single View Reconstruction Results [Mescheder2018]





Test on Real Images [Mescheder2018]



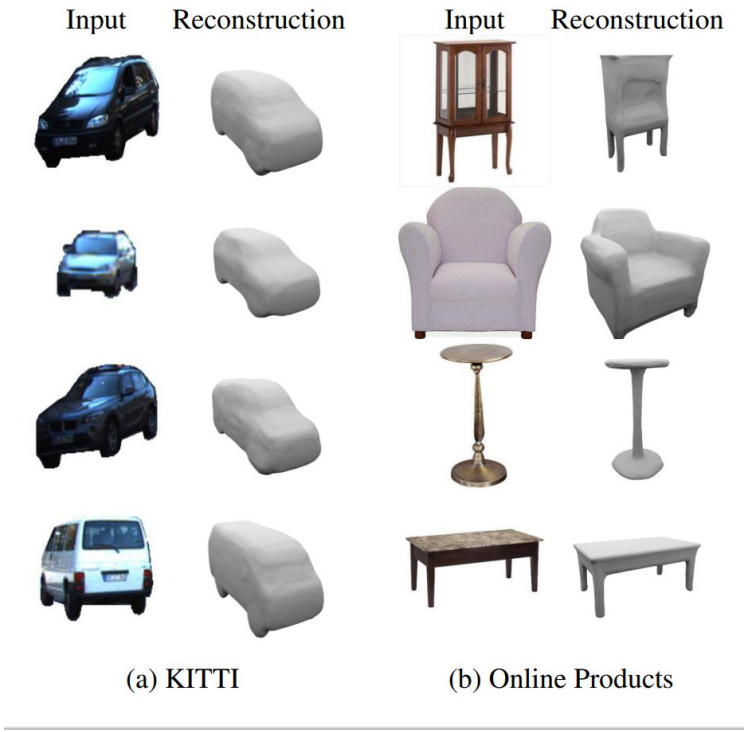
Interpolation Results [Park2019]

Decoder
D

Limitations so far - SDF

Reconstructed models are too smooth

Decoder
D



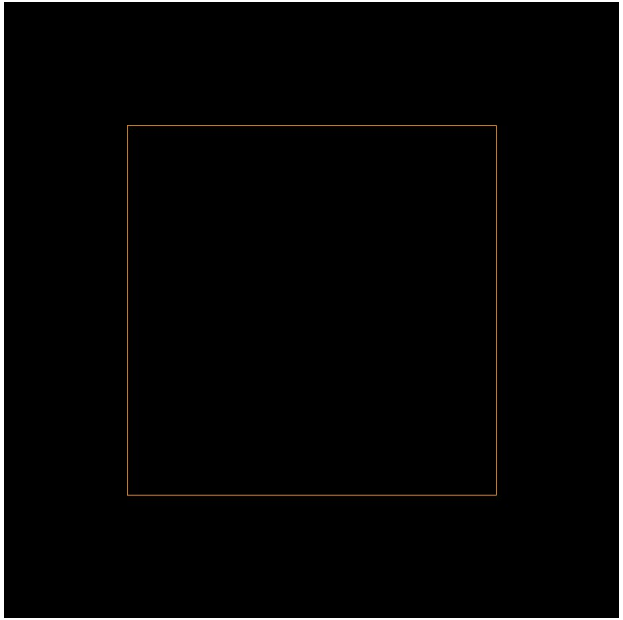
Limitations so far - SDF

Decoder
D

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

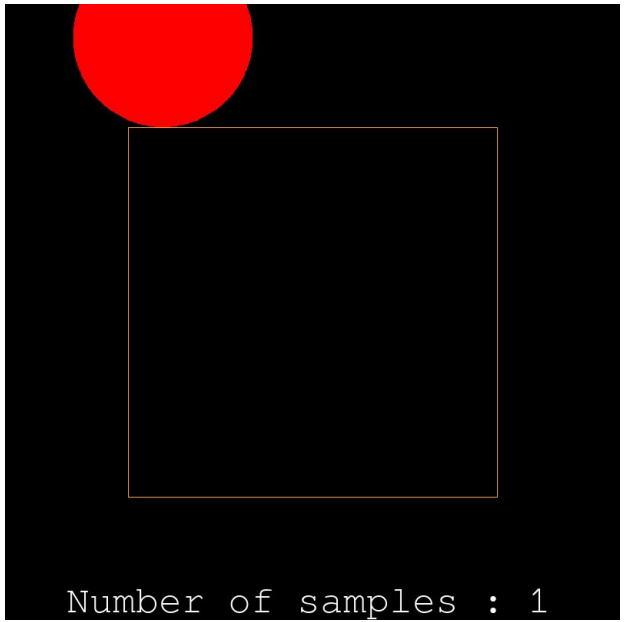


Limitations so far - SDF

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

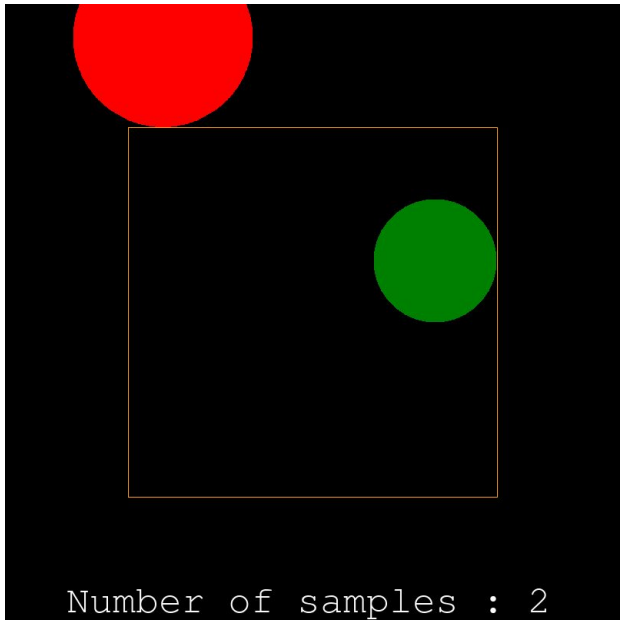


Limitations so far - SDF

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

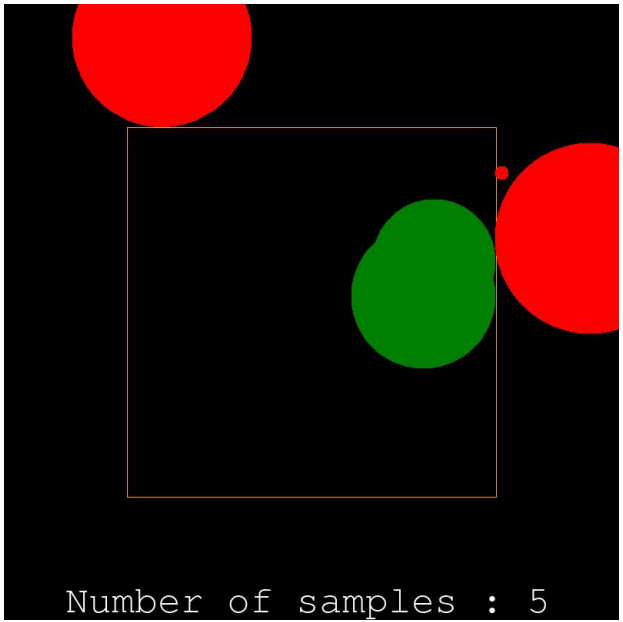


Limitations so far - SDF

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

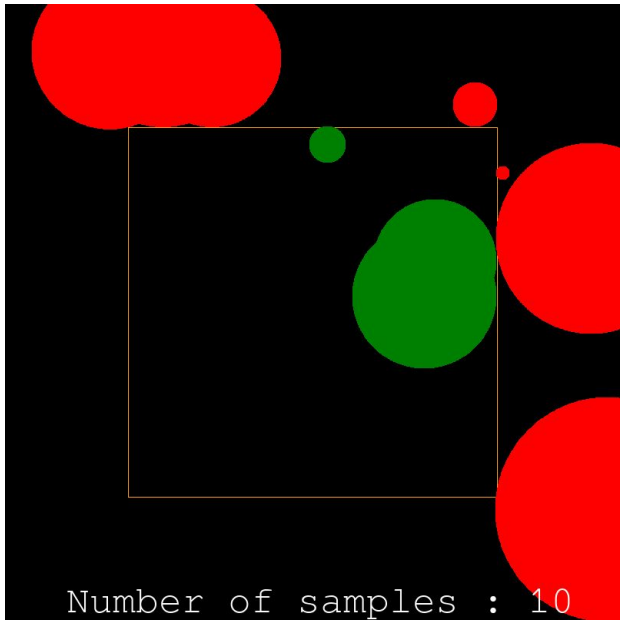


Limitations so far - SDF

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

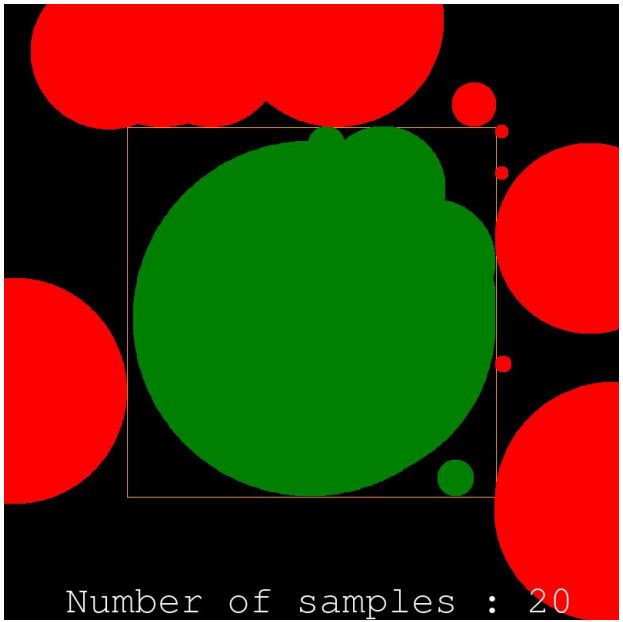


Limitations so far - SDF

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

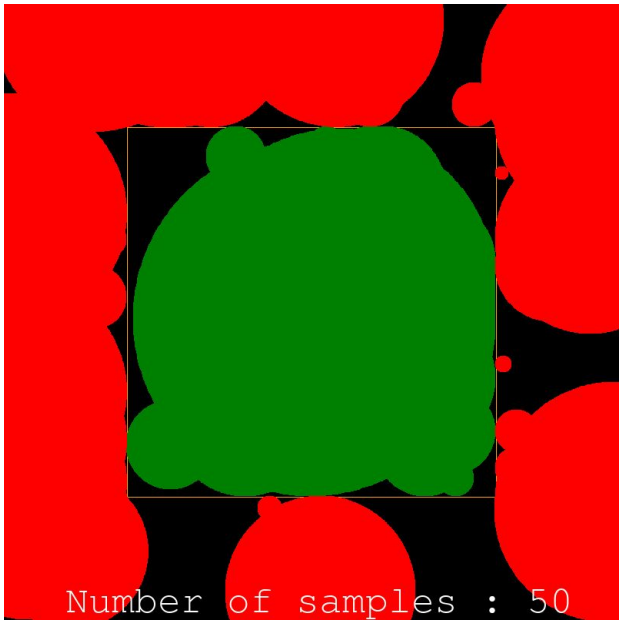


Limitations so far - SDF

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

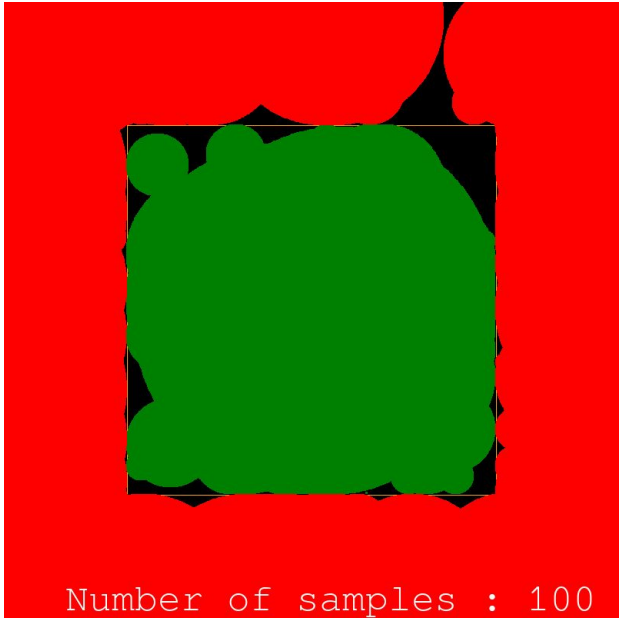


Limitations so far - SDF

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

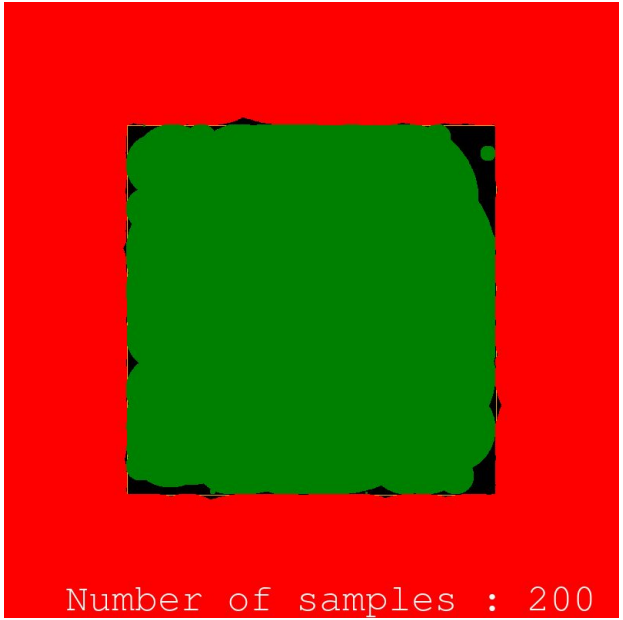


Limitations so far - SDF

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

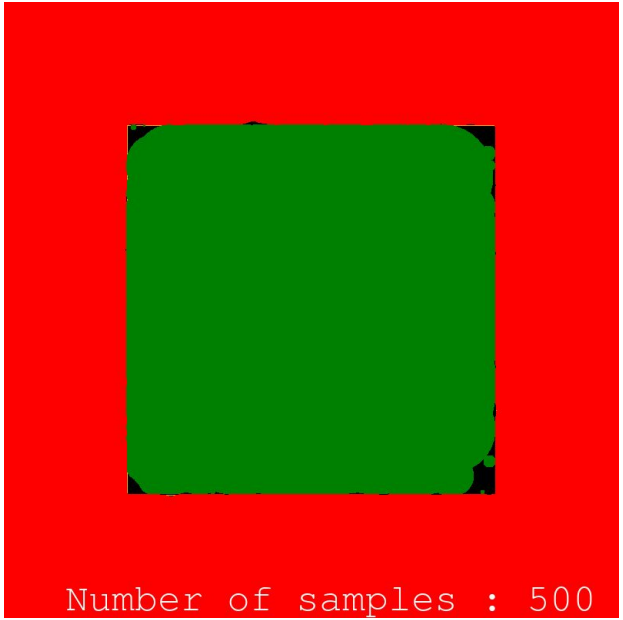


Limitations so far - SDF

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

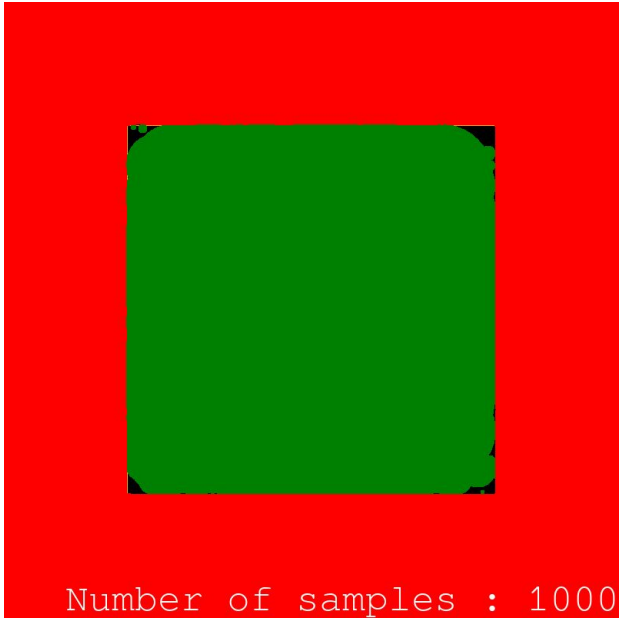


Limitations so far - SDF

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

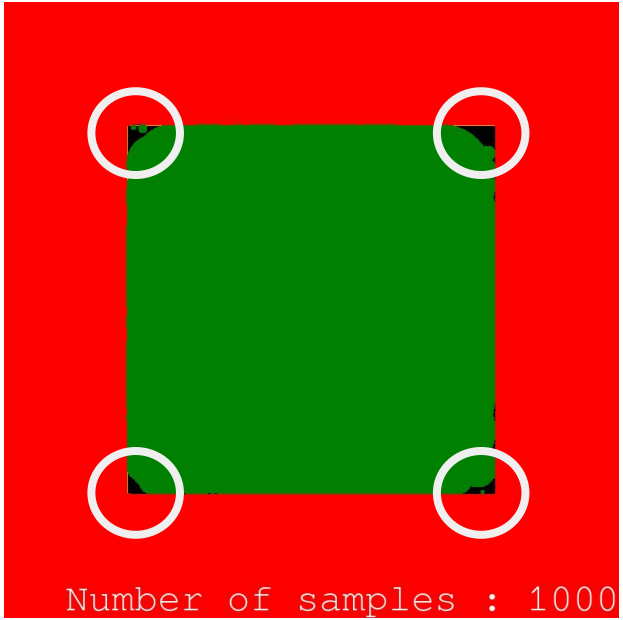
Decoder
D

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

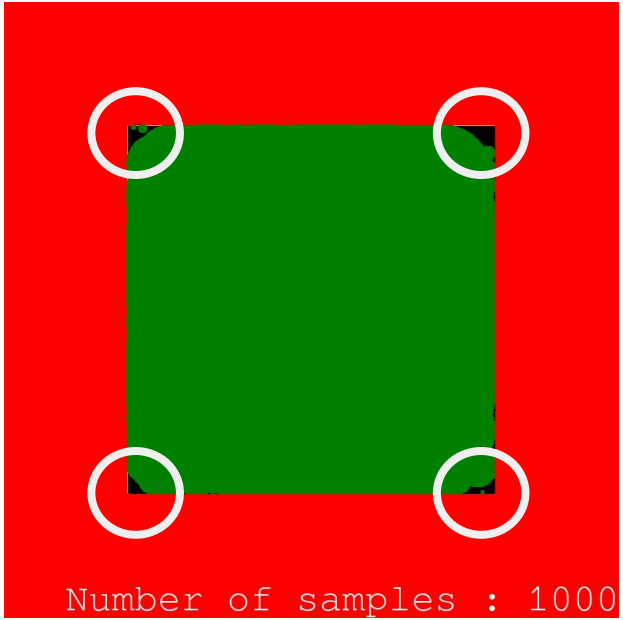
Reconstructed models are too smooth - Possible explanation : Monte-Carlo sampling

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**

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)

Fitting geometric primitives to a 3D shape

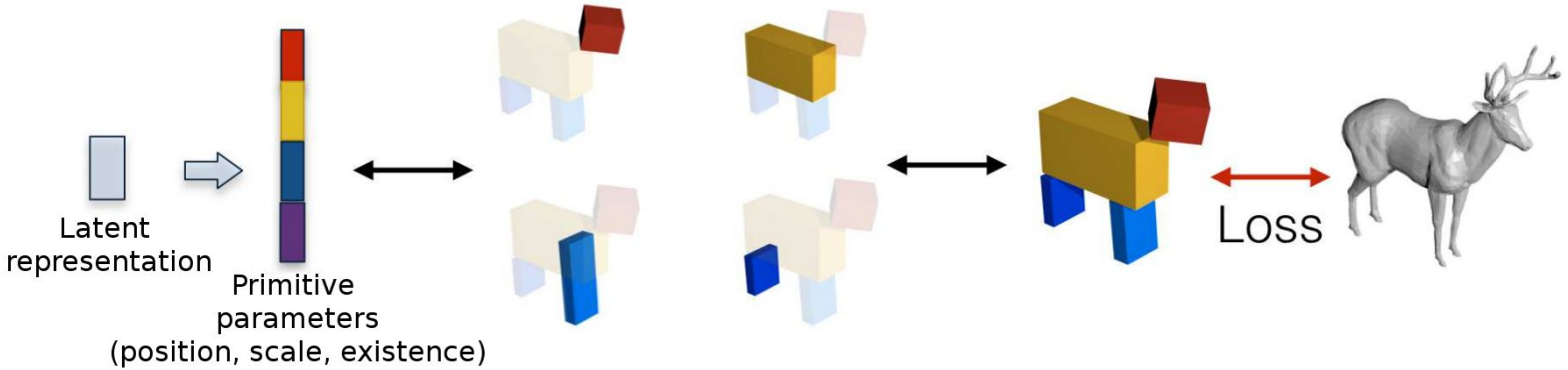
Everything in nature takes its form from the sphere, the cone and the cylinder.
- Paul Cezanne.

Motivations :

- Parsimony of description
- Helps finding structures in images for abstraction or animation
- In the case of geometric object, helps capturing details (sharp angles)

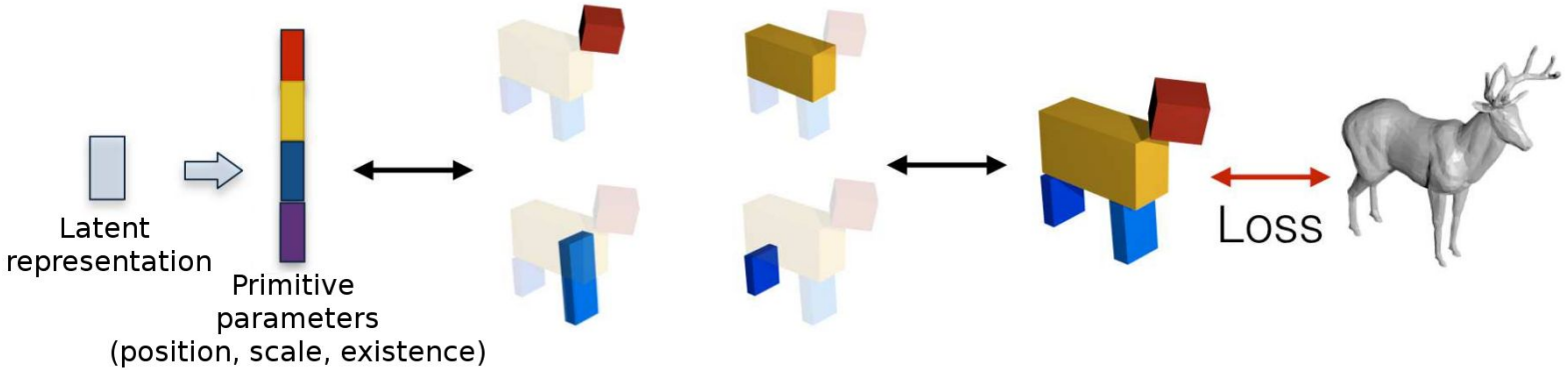
Learn 3D reconstruction with cuboids [Tulsiani2017]

Unsupervised method for fitting cuboid primitives



Learn 3D reconstruction with cuboids [Tulsiani2017]

Unsupervised method for fitting cuboid primitives



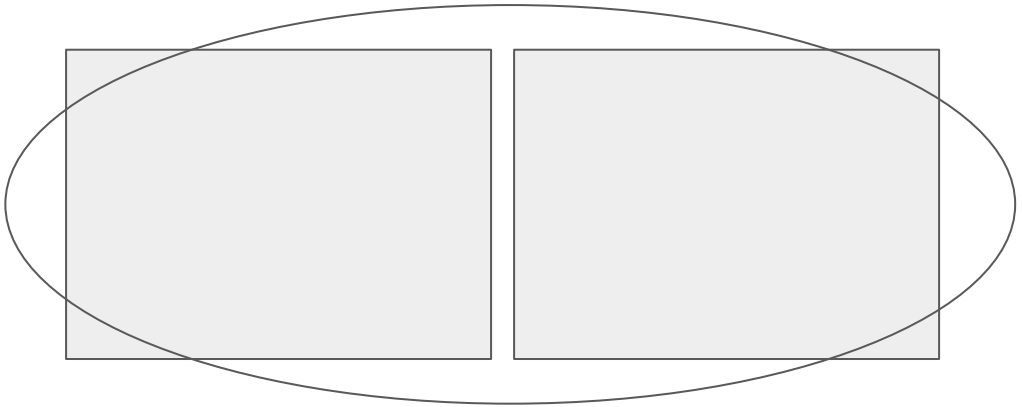
Challenges :

1. Position the cuboids
2. Estimating the amount of cuboids to predict

Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss : Chamfer ?

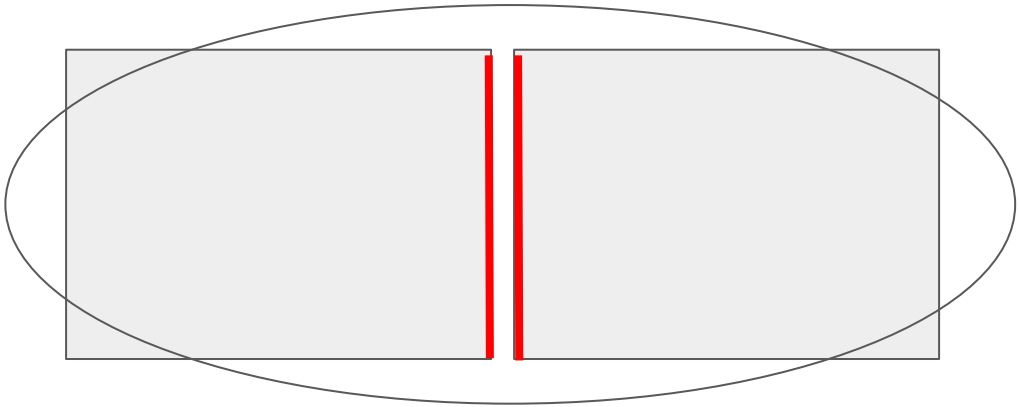
Problem !



Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss : Chamfer ?

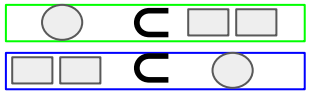
Problem !



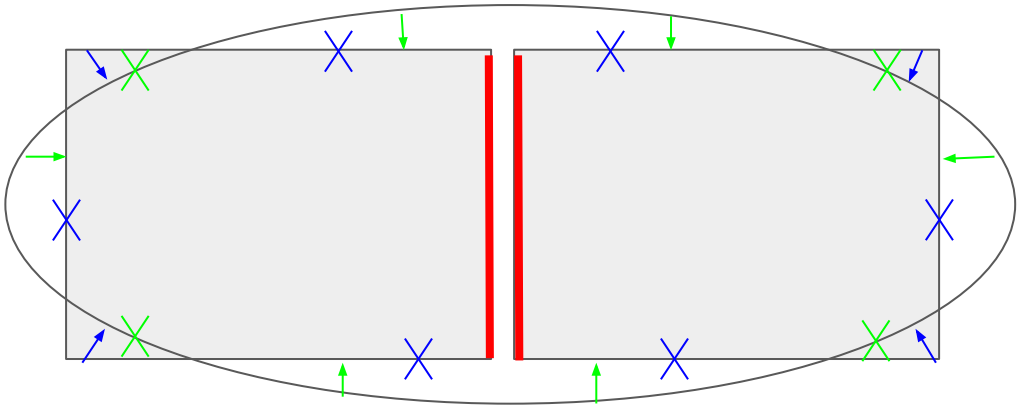
➔ Points sampled on ——— increase the Chamfer distance

Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss : Chamfer ?



Problem !



➔ Points sampled on ▬ increase the Chamfer distance

➔ **Solution** :

- Among points sampled on ◯ , we discard points which are inside ▭▭
- Among points sampled on ▭▭, we discard points which are inside ◯

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]

Decoder
D

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 p_m

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 p_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

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

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 p_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 p_m"

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 p_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 p_m"

How to back-propagate through an expectation ?

Estimating the amount of cuboids to predict [Tulsiani2017]

Back-propagate through an expectation [Williams1992]

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

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_θ parametrized by θ.
Let $f : \mathcal{X} \to \mathbb{R}$
We want to evaluate

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

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_θ parametrized by θ.
Let $f : \mathcal{X} \to \mathbb{R}$
We want to evaluate

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

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

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_θ parametrized by θ.
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"

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_θ parametrized by θ.
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"

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

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_θ parametrized by θ.
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_θ parametrized by θ.
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 $(X_n)_{n\in\{1,N\}} \sim p_\theta$

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

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_θ parametrized by θ.
Let $f : \mathcal{X} \to \mathbb{R}$

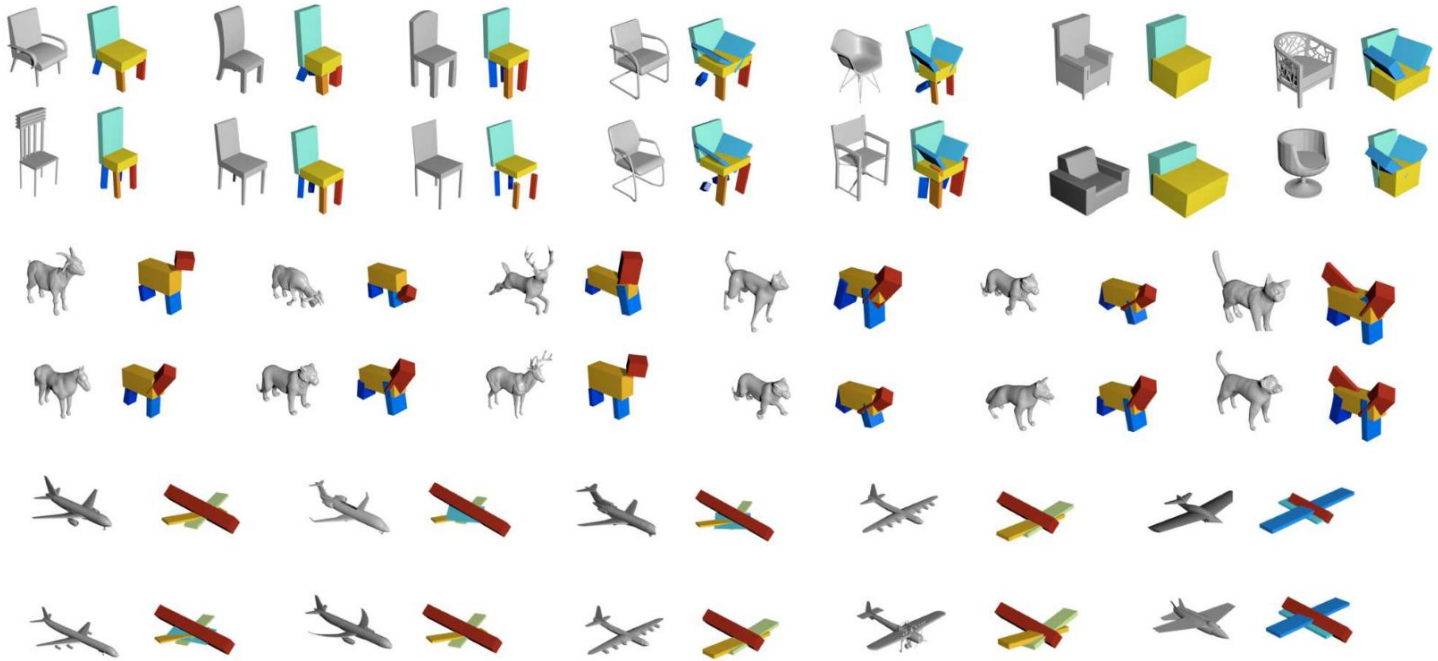
$$\frac{\partial}{\partial\theta}\mathbb{E}[f(X)] \approx \frac{1}{N}\sum_{n=1}^{N}\left[f(X_n)\frac{\partial}{\partial\theta}\log(p_\theta(X_n))\right]$$

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

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)

Very Modular Framework ! [Tulsiani2017]

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

PointCloud (Voxelized)

...

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**)

3D Object **Y**

Very Modular Framework ! [Groueix2018]

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)

Very Modular Framework ! [Choy2016, Tatarchenko2017]

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)

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 : SphericalCNNs

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

- ★ **PointNet [Qi2017] :** Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proc. Computer Vision and Pattern Recognition (CVPR), IEEE 1.2 (2017): 4.
- ★ [**Wang2018**] : Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2018). Dynamic graph CNN for learning on point clouds. arXiv preprint arXiv:1801.07829.
- ★ [**Jiang2018**] : Jiang, Mingyang, Yiran Wu, and Cewu Lu. "Pointsift: A sift-like network module for 3d point cloud semantic segmentation." arXiv preprint arXiv:1807.00652 (2018).
- ★ [**Klokov2017**] : Klokov, Roman, and Victor Lempitsky. "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models." *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017.
- ★ **PointNet++** [**Qi2017b**] : Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." *Advances in Neural Information Processing Systems*. 2017.
- ★ [**Guerrero2017**]: Guerrero, Paul, et al. "PCPNet Learning Local Shape Properties from Raw Point Clouds." *Computer Graphics Forum*. Vol. 37. No. 2. 2018.
- ★ [**Landrieu2018**] : Landrieu, Loic, and Martin Simonovsky. "Large-scale point cloud semantic segmentation with superpoint graphs." *arXiv preprint arXiv:1711.09869* (2017).
- ★ [**Yi2016**] : Yi, Li, et al. "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation." *CVPR*. 2017.

Spherical representations

- ★ [**Esteves2018**] : Esteves, Carlos, et al. "Learning so (3) equivariant representations with spherical cnns." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- ★ [**Cohen2018**] : Cohen, Taco S., et al. "Spherical CNNs." *arXiv preprint arXiv:1801.10130* (2018).

Bibliography : Encoder

Graph

- [**Simonovsky2017**] : Simonovsky, Martin, and Nikos Komodakis. "Dynamic edge-conditioned filters in convolutional neural networks on graphs." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017.

Voxels

- **[Riegler2017] :** Riegler, G., Ulusoy, A. O., Bischof, H., & Geiger, A. (2017, October). Octnetfusion: Learning depth fusion from data. In 3D Vision (3DV), 2017 International Conference on (pp. 57-66). IEEE.
- **3d-r2n2 [Choy2016]** : C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In Proc. of the European Conf. on Computer Vision (ECCV), 2016.
- **Voxnet [Maturana2015]** : D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS), 2015.
- **Octnet [Riegler2017] :** Riegler, Gernot, Ali Osman Ulusoy, and Andreas Geiger. "Octnet: Learning deep 3d representations at high resolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Vol. 3. 2017.

Images

- **Resnet [He2015]** : He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- **[Qi2016]** : C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.

Bibliography : Decoder

Points

- ★ **PointSetGen [Fan2017]** : Fan, Haoqiang, Hao Su, and Leonidas J. Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image." CVPR. Vol. 2. No. 4. 2017.

Voxels

- ★ **OGN [Tatarchenko2017]** : Tatarchenko, Maxim, Alexey Dosovitskiy, and Thomas Brox. "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs." Proc. of the IEEE International Conf. on Computer Vision. Vol. 2. 2017.
- ★ **[Delanoy2017]** : Delanoy, Johanna, et al. "What you sketch is what you get: 3D sketching using multi-view deep volumetric prediction." arXiv preprint arXiv:1707.08390 (2017).

Surfaces

- ★ **[Groueix2018]** : Groueix, T., Fisher, M., Kim, V., Russell, B., and Aubry, M. (2018, June). AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In CVPR 2018.
- ★ [**Groueix2018b**] : Groueix, Thibault, et al. "3D-CODED: 3D Correspondences by Deep Deformation." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

Mesh

- ★ [**Simonovsky2017**] : Simonovsky, Martin, and Nikos Komodakis. "Dynamic edge-conditioned filters in convolutional neural networks on graphs." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017.

Depth maps

- ★ [**Tulsiani2018**] : Tulsiani, Shubham, Richard Tucker, and Noah Snavely. "Layer-structured 3d scene inference via view synthesis." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

Signed Distance Function

- ★ [**Chen2018**] : Chen, Zhiqin, and Hao Zhang. "Learning Implicit Fields for Generative Shape Modeling." arXiv preprint arXiv:1812.02822(2018).
- ★ [**Park2019**] : Park, Jeong Joon, et al. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation." arXiv preprint arXiv:1901.05103 (2019).
- ★ [**Mescheder2018**] : Mescheder, Lars, et al. "Occupancy Networks: Learning 3D Reconstruction in Function Space." arXiv preprint arXiv:1812.03828 (2018).

Bibliography : Decoder

Geometric primitives

★ **[Tulsiani2017]** : Tulsiani, Shubham, et al. "Learning shape abstractions by assembling volumetric primitives." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017.

Bibliography

Optimal Transport

- ★ **[Kuhn1955]** : Kuhn, Harold W. "The Hungarian method for the assignment problem." 50 Years of Integer Programming 1958-2008. Springer, Berlin, Heidelberg, 2010. 29-47.
- ★ **[Cuturi2013]** : Cuturi, Marco. "Sinkhorn distances: Lightspeed computation of optimal transport." Advances in neural information processing systems. 2013.
- ★ **[Altschuler2017]** : Altschuler, Jason, Jonathan Weed, and Philippe Rigollet. "Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration." Advances in Neural Information Processing Systems. 2017.
- ★ **[Bertsekas1988]** : Bertsekas, Dimitri P. "The auction algorithm: A distributed relaxation method for the assignment problem." Annals of operations research 14.1 (1988): 105-123.

Datasets

- ★ [**Wu2015**] : Wu, Zhirong, et al. "3d shapenets: A deep representation for volumetric shapes." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
- ★ [**Bogo2014**] : Bogo, F., Romero, J., Loper, M., & Black, M. J. (2014). FAUST: Dataset and evaluation for 3D mesh registration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3794-3801).
- ★ [**Bogo2017**] : Bogo, F., Romero, J., Pons-Moll, G., & Black, M. J. (2017, July). Dynamic FAUST: Registering human bodies in motion. In IEEE Conf. on Computer Vision and Pattern Recognition (Vol. 6).
- ★ [**Song2017**] : Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., & Funkhouser, T. (2017, July). Semantic scene completion from a single depth image. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on (pp. 190-198). IEEE.
- ★ [**Sun2018**] : Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., Xue, T., ... & Freeman, W. T. (2018, April). Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2974-2983).
- ★ [**Lim2013**] : Lim, J. J., Pirsiavash, H., & Torralba, A. (2013). Parsing ikea objects: Fine pose estimation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2992-2999).

Marching Cubes

- ★ [**Liao2018**] : Y. Liao, S. Donne, and A. Geiger. Deep marching cubes: Learning explicit surface representations. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2018
- ★ [**Lorensen1987**] : W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In ACM Trans. on Graphics (SIGGRAPH), 1987.

Bibliography

Other

- ★ **[Furukawa2009]** : Furukawa, Yasutaka, et al. "Manhattan-world stereo." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.
- ★ **[Williams1992]** : Williams, R.J. Mach Learn (1992) 8: 229. https://doi.org/10.1007/BF00992696

Additional Material

PointClouds Analysis Motivation

Source : http://graphics.stanford.edu/courses/cs468-17-spring/schedule.html

- **Robot Perception**

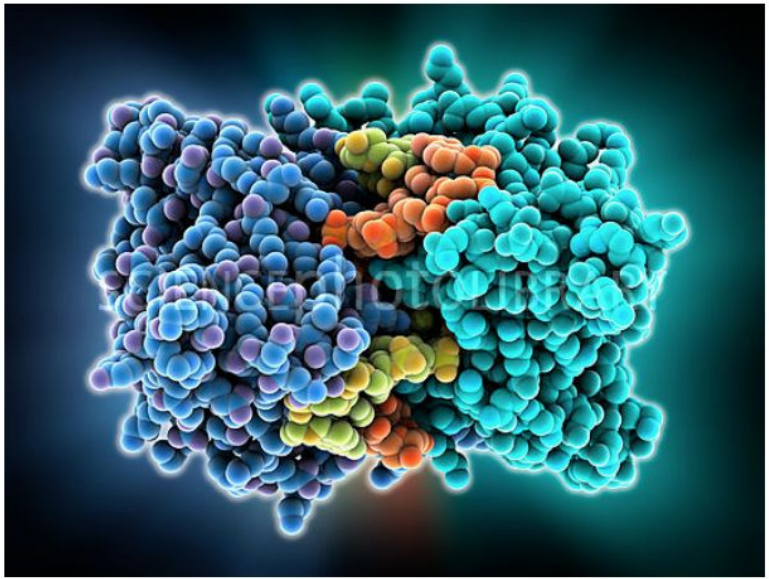
What and where are the objects in a LiDAR scanned scene?

https://3dprint.com/116569/self-driving-cars-privacy/

PointClouds Analysis Motivation

- **Molecular Biology**

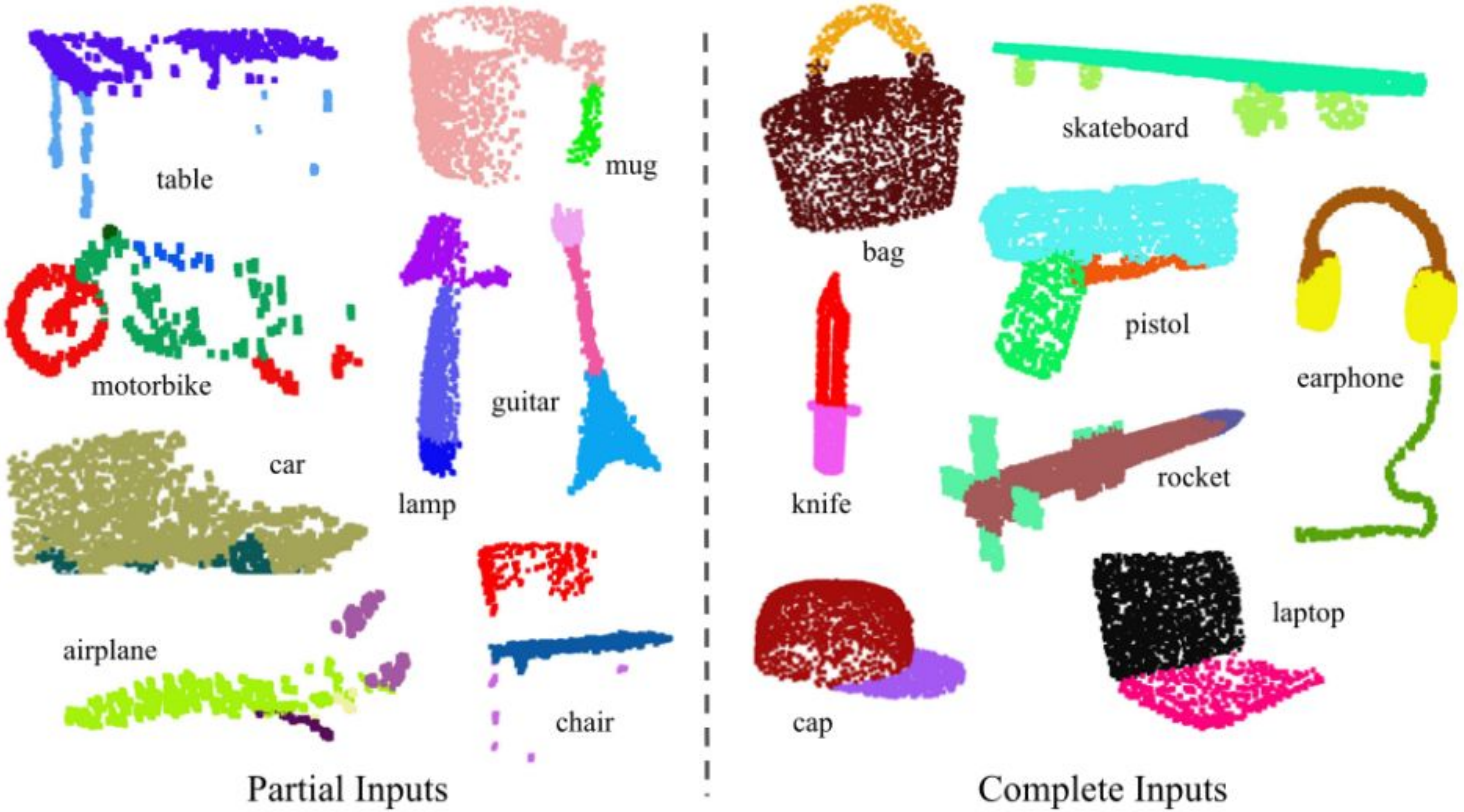
Can we infer an enzyme's category (reactions they catalyze) from its structure?



EcoRV restriction enzyme molecule, LAGUNA DESIGN/SCIENCE PHOTO LIBRARY

PointNet Results : object semantic segmentation

Credit [**Qi2017**]

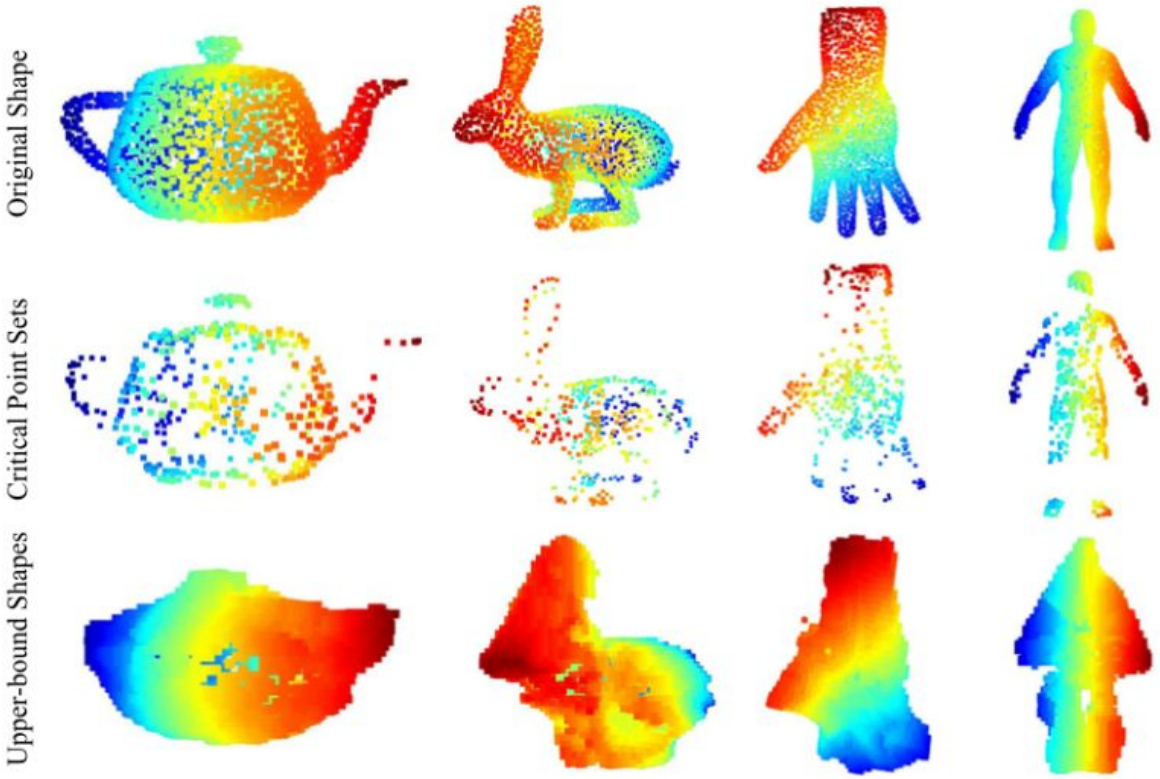


PointNet Results : scene semantic segmentation

Credit [**Qi2017**]

PointNet Analysis : Critical and Upper bound Point Set

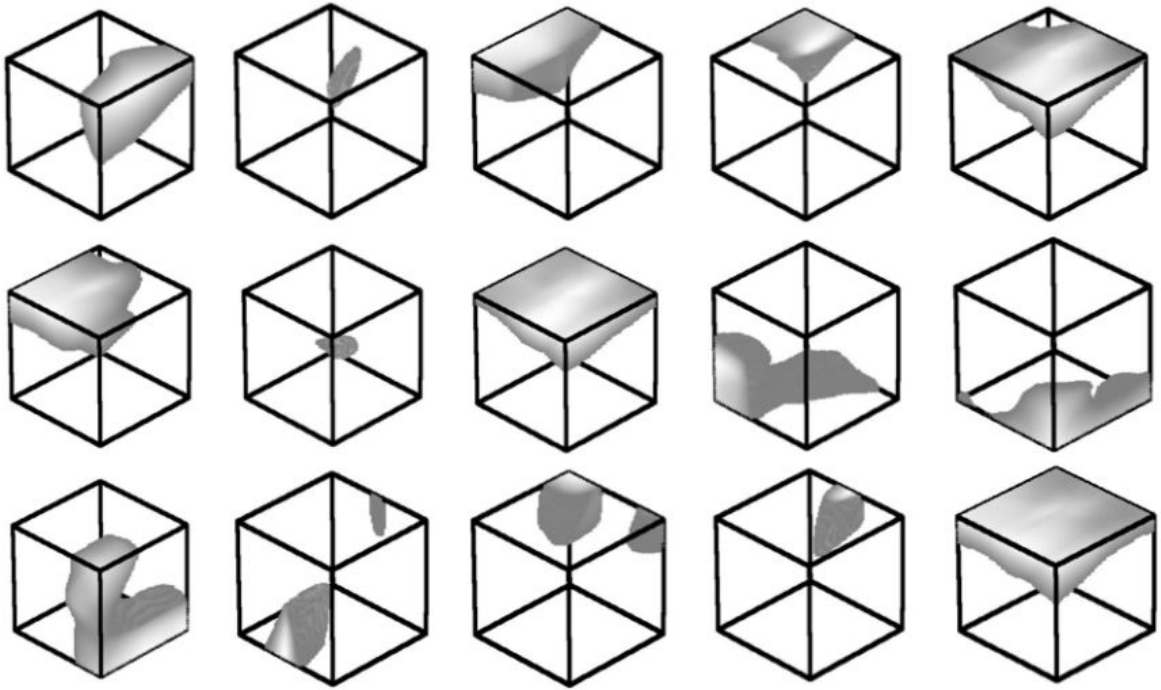
Credit [**Qi2017**]



PointNet Analysis : features activation

Credit [**Qi2017**]

➔ Find the top-K points in a dense volumetric grid that activates neuron X.



	Analyse	Generation
RGBD	CNNs (resnet)	CNNs (resnet)
Mesh	SyncSpecNet Graph CNNs	AtlasNet Neural renderer RenderNet
Image-based Methods	Dosovitsky et al, ECCV 2016	SurfNet
Voxel Based Methods	3D-r2n2 OctNet Hierarchical Surface Prediction	3D-r2n2 Hierarchical Surface Prediction Octree Generative Networks
Point Based Methods	PointNet, PointNet++ SuperPoints Graph (large scale) PCPNet	PointSetGen
Primitives	Shape Abstraction	Shape Abstraction Supervised Fitting of Geometric Primitives to 3D Point Clouds

Datasets

	Objects/ Scenes	Semantic	Correspondences	Synthetic/Real	paired with real 2D images
ShapeNet	Objects	x		Synthetic	
ModelNet	Objects	x		Synthetic	
Pix3D	Objects			Real	x
Faust	Humans		x	Real	
SunCG	Scenes	x		Synthetic	
Stanford2D/3D	Scenes	x			

Shapenet

https://www.shapenet.org/

ShapeNetCore : 55 common object categories with about 51,300 unique 3D models

ShapenetSem : ShapeNetSem is a smaller, more densely annotated subset consisting of 12,000 models spread over a broader set of 270 categories.

ModelNet : http://modelnet.cs.princeton.edu/

ModelNet contains 127,915 CAD models, 662 categories

Airplane	Bathtub	Bed	Bench	Book shelf	Bottle	Bowl	Car
8712	1872	7380	2316	8064	5220	1008	3564
Chair	Cone	Curtain	Cup	Desk	Door	Dresser	Flower Pot
11868	2244	1188	1896	3432	1548	3432	2028
Glass Box	Guitar	Keyboard	Lamp	Laptop	Mantel	Monitor	Person
3252	3060	1980	1728	2028	4608	6780	3432
Night Stand	Piano	Plant	Radio	Range hood	Sink	Sofa	Stairs
1296	3972	4080	1488	2580	1776	9360	1728
Table	Stool	Tent	Toilet	TV Stand	Vase	Wardrobe	Xbox
1320	5904	2196	5328	4404	6900	1284	1476

Subset: ModelNet10

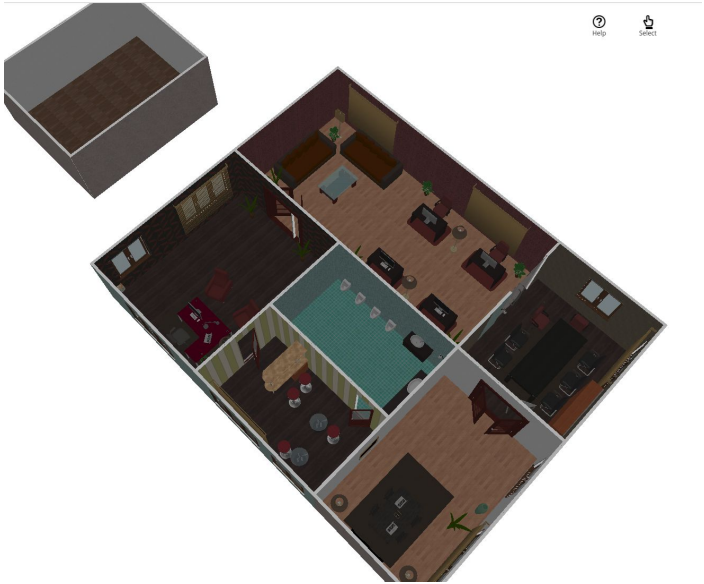
				
Bathtub	Bed	Chair	Desk	Dresser
156	615	989	286	286
				
Monitor	NightStand	Sofa	Table	Toilet
565	286	780	490	444

credit Ayesha Ahmad Thesis

Princeton Shape Benchmark

SunCG http://suncg.cs.princeton.edu/

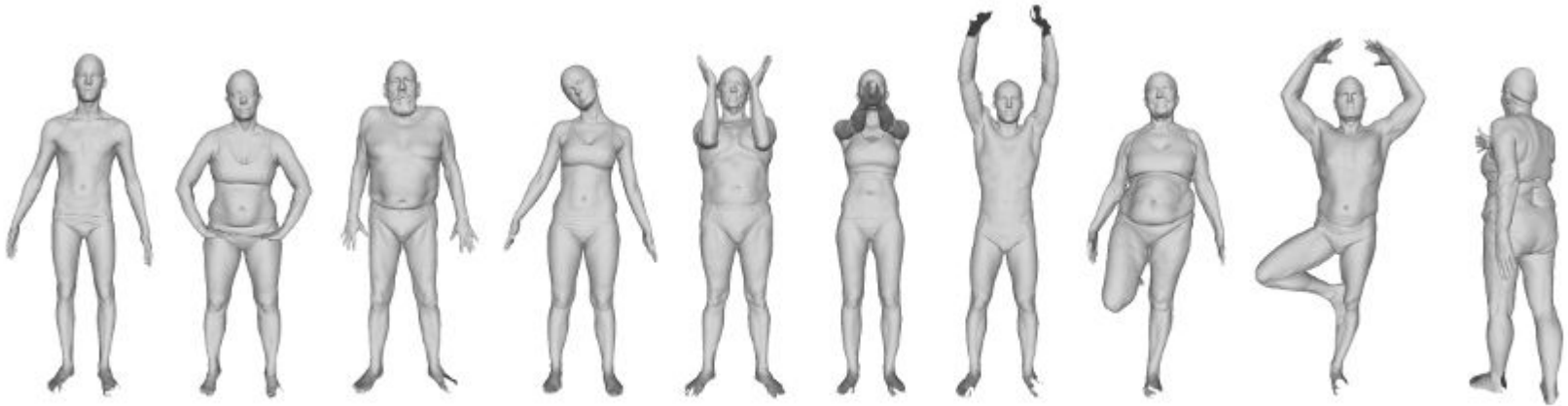
The dataset contains over 45K different scenes with manually created realistic room and furniture layouts. All of the scenes are semantically annotated at the object level. (2000 objects)



FAUST : http://faust.is.tue.mpg.de/overview

Train : 100 shapes with dense correspondences

Test : 200 shapes with an online benchmark



Overview of the 10 poses in the training set.

Dynamic FAUST : http://dfaust.is.tue.mpg.de/

40k samples : 4D data, moving human shapes over time, registered and aligned.

ObjectNet3D : images and shapes

Pascal 3D+ : images and shapes

IKEA fine-grained pose annotations for shapes (2013)

http://ikea.csail.mit.edu/

759 (real) images and 219 3D models

Pix3D extends IKEA dataset

Pix3D has 395 3D shapes of nine object categories. Each shape associates with a set of real images, capturing the exact object in diverse environments. Further, the 10,069 image-shape pairs have precise 3D annotations, giving pixel-level alignment between shapes and their silhouettes in the images.
Subset from IKEA : 219 GT IKEA meshes + 14600 images from web search
Subset added : 209 Scanned object (depth fusion algo) with 2313 pictures
After merge and refinement : **395 shapes with 10069 aligned images**

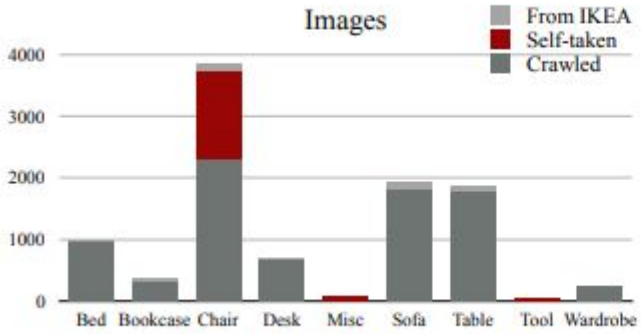


Figure 4: The distribution of images across categories

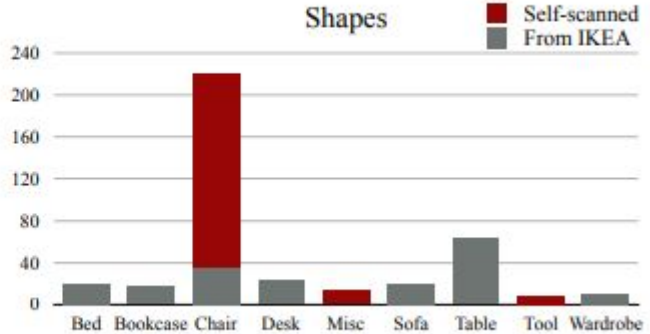


Figure 5: The distribution of shapes across categories

SHREC / SCAPE

Stanford 2D/3D

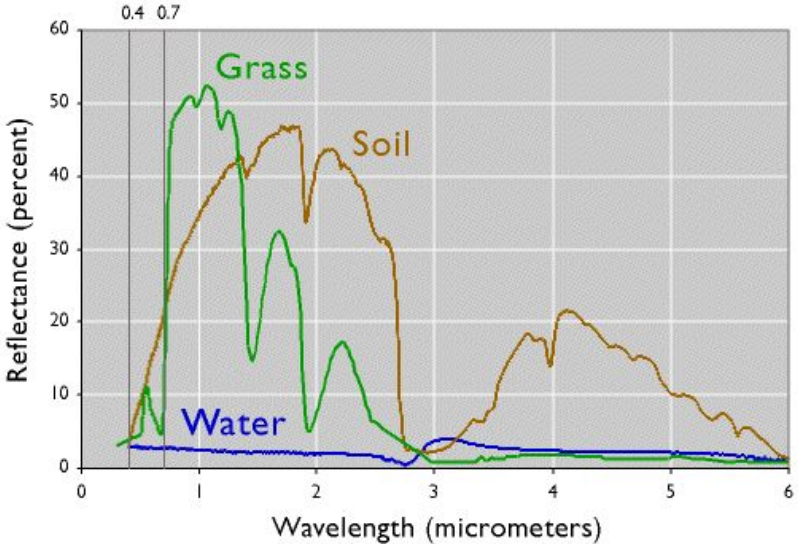
Tanks and Temples

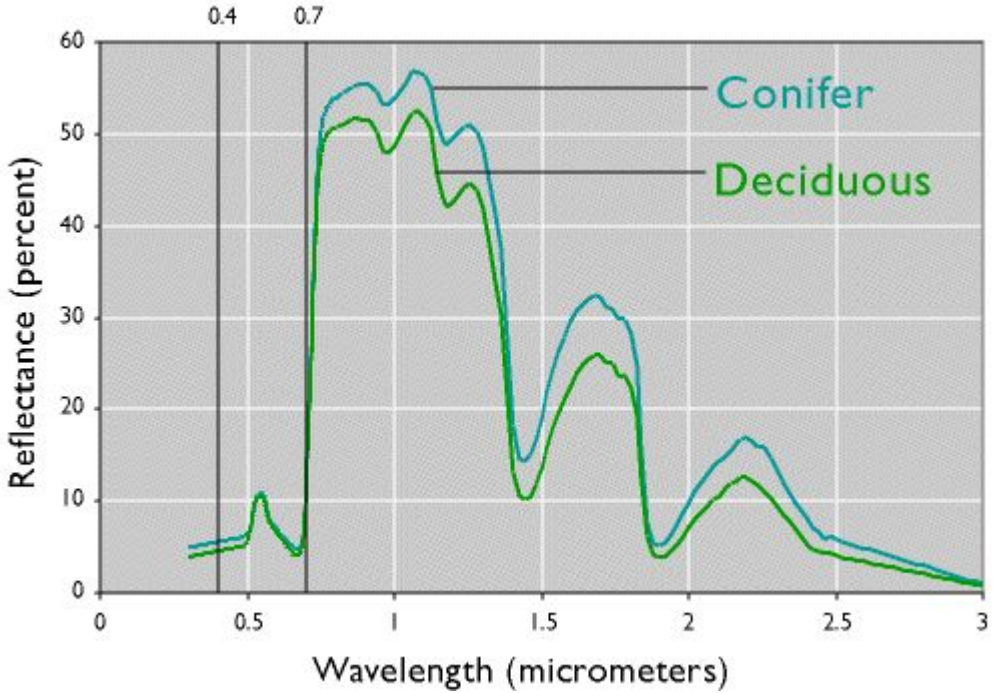
Semantic 3D

Dataset of Superpoint Graph [Landrieu2018]

Hyperspectral Cameras

Exploring beyond the visible spectrum provides richer information





Source : [Aster Spectral Library]

Concrete example

In this image with artificial colors, the red corresponds to the light emitted in the near infrared spectrum. Chlorophyll emits strongly in this part of the spectrum[1]: this clearly separates the vegetation from the rest of the image.

Constraints and examples

A tradeoff between spatial resolution and spectral resolution (from Shanon Theorem).

Satellite
Pleiade : camera optique (multi-spectral) $\Delta\lambda$ = 120nm , Δx=2m
Hyperion : hyperspectral $\Delta\lambda$ = 20nm , Δx=30m
Enmap : hyperspectral $\Delta\lambda$ = 10nm , Δx=30m

Airborn
Aviris : Hyperspectral. At 20km elevation Δx=20m (price : 10k)

Cool Stuff :
Sentinel takes 1 picture of the whole planet every 10 days and open-source the data.
Sentinel 1 : Radar Δx=10m
Sentinel 2 : Multi-Spectral Δx=10m

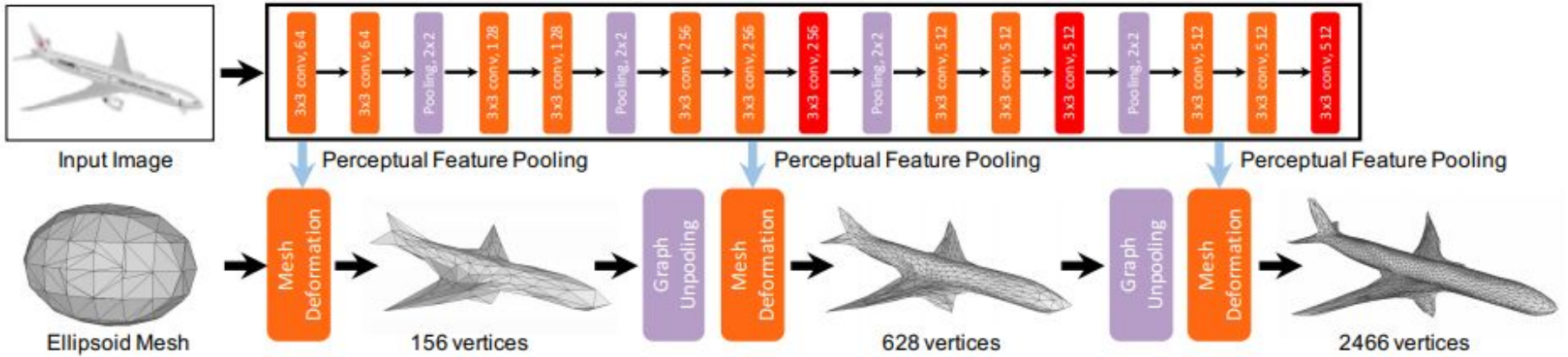
Donnée IGN

Ortho-Images du territoire français depuis 1 siècle !

https://remonterletemps.ign.fr/

[**Wang2018**] : Wang, Nanyang, et al. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." *arXiv preprint arXiv:1804.01654* (2018).

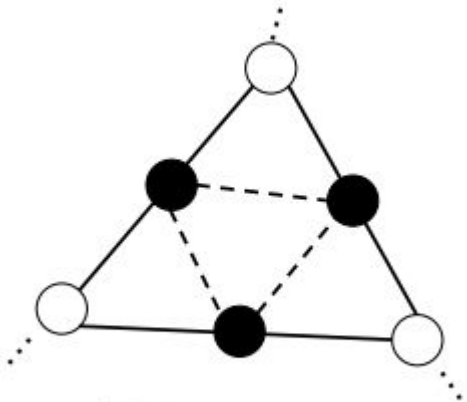
Pixel2Mesh



++ you control the resolution of the graph so you can easily take a **coarse to fine** approach (increased stability)
-- You are limited by the template (genus 0 shapes)

Graph Unpooling, credit [Wang2018]

No learning, just graphics



(a) Graph Unpooling

Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss

Distance Field of a point *p* to an object *O*
(Evaluates to 0 inside object *O*)

$$\mathcal{C}(p;O) = \min_{p' \in O} \|p - p'\|_2$$

Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss

Distance Field of a point *p* to an object *O*

$$\mathcal{C}(p;O) = \min_{p' \in O} \|p - p'\|_2$$

Is my target object *O* inside my predicted shape $\bigcup_m \bar{P}_m$ formed by m primitives ?

$$L_1(\bigcup_m \bar{P}_m, O) = \mathbb{E}_{p \sim S(O)} \|\mathcal{C}(p; \bigcup_m \bar{P}_m)\|^2$$

p is sampled on *S(O),* the surface of the target object O

Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss

Distance Field of a point *p* to an object *O*

$$\mathcal{C}(p;O) = \min_{p' \in O} \|p - p'\|_2$$

Is my target object *O* inside my predicted shape $\bigcup_m \bar{P}_m$ formed by m primitives ?

$$L_1(\bigcup_m \bar{P}_m, O) = \mathbb{E}_{p \sim S(O)} \|\mathcal{C}(p; \bigcup_m \bar{P}_m)\|^2$$

p is sampled on *S(O),* the surface of the target object O

Is my predicted shape $\bigcup_m \bar{P}_m$ inside my target object *O* ?

$$L_2(\bigcup_m \bar{P}_m, O) = \sum_m \mathbb{E}_{p \sim S(\bar{P}_m)} \|\mathcal{C}(p; O)\|^2$$

p is sampled on the surface of the m-th predicted cuboid

Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss

Distance Field of a point *p* to an object *O*

$$\mathcal{C}(p;O) = \min_{p' \in O} \|p - p'\|_2$$

Is my target object *O* inside my predicted shape $\bigcup_m \bar{P}_m$ formed by m primitives ?

$$L_1(\bigcup_m \bar{P}_m, O) = \mathbb{E}_{p \sim S(O)} \|\mathcal{C}(p; \bigcup_m \bar{P}_m)\|^2$$

p is sampled on *S(O)*, the surface of the target object O

Is my predicted shape $\bigcup_m \bar{P}_m$ inside my target object *O* ?

$$L_2(\bigcup_m \bar{P}_m, O) = \sum_m \mathbb{E}_{p \sim S(\bar{P}_m)} \|\mathcal{C}(p; O)\|^2$$

p is sampled on the surface of the m-th predicted cuboid

$L_1 + L_2$ is an adaptation of the Chamfer distance optimized for cuboids