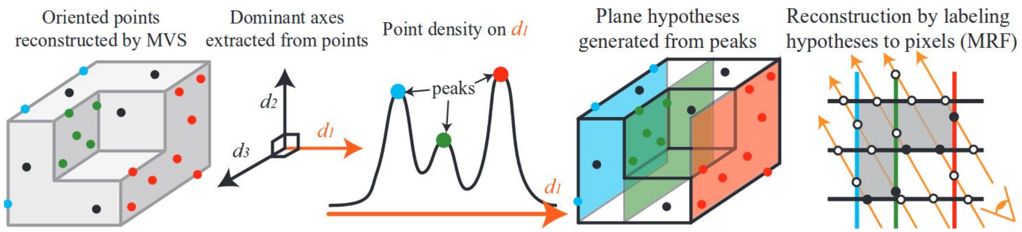
Deep Learning for 3D
Toward Surface Generation

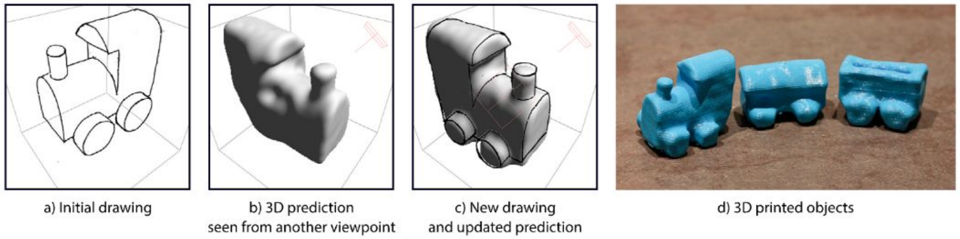
Thibault GROUEIX, Pierre-Alain LANGLOIS

Why learn ?

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



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



Data types

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

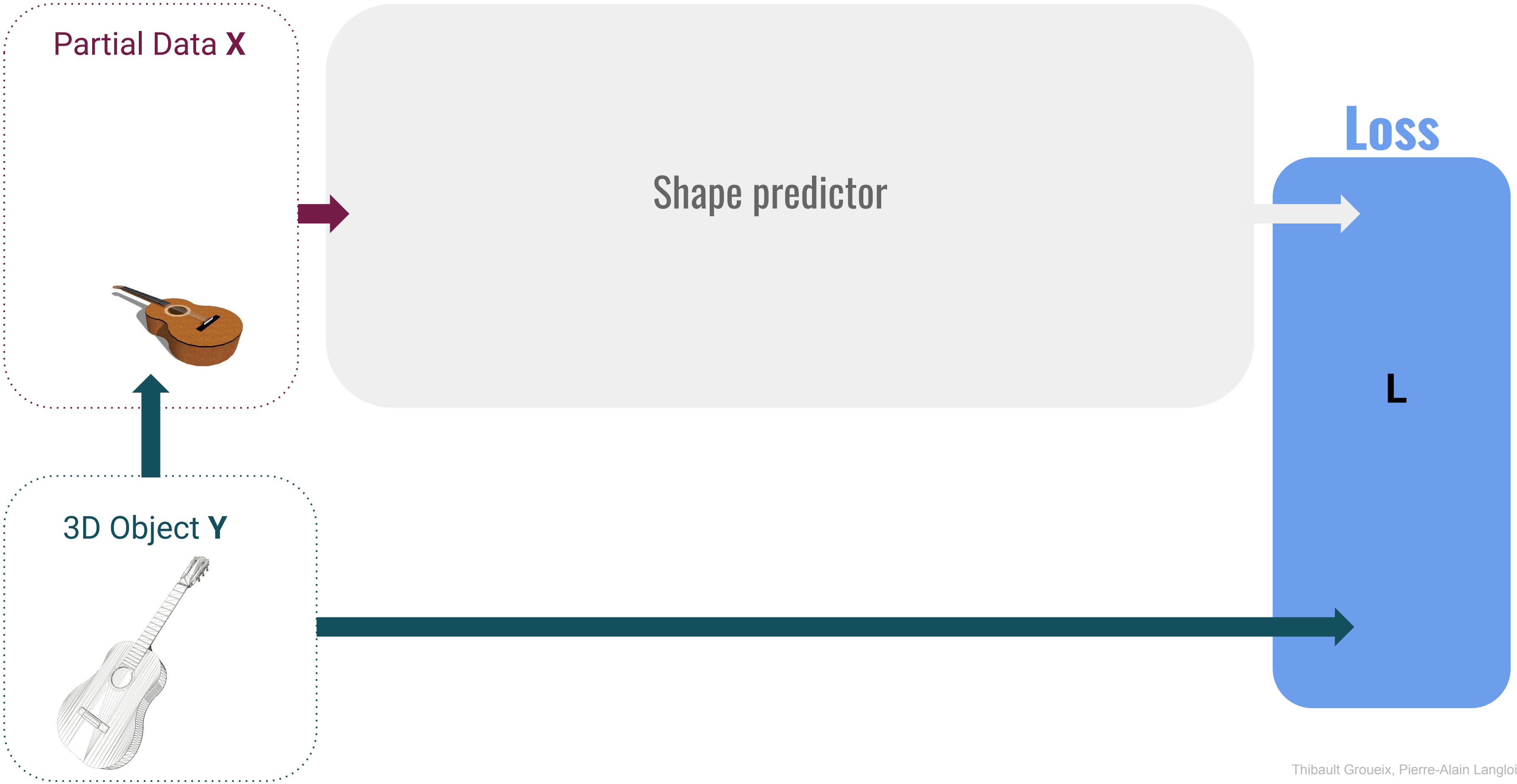
RGB Image(s)

RGBD Image(s)

PointCloud

Thibault Groueix, Pierre-Alain Langlois, 2019

Typical learning framework based on synthetic data



Training setup for 3D reconstruction

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

PointCloud (Voxelized)

...

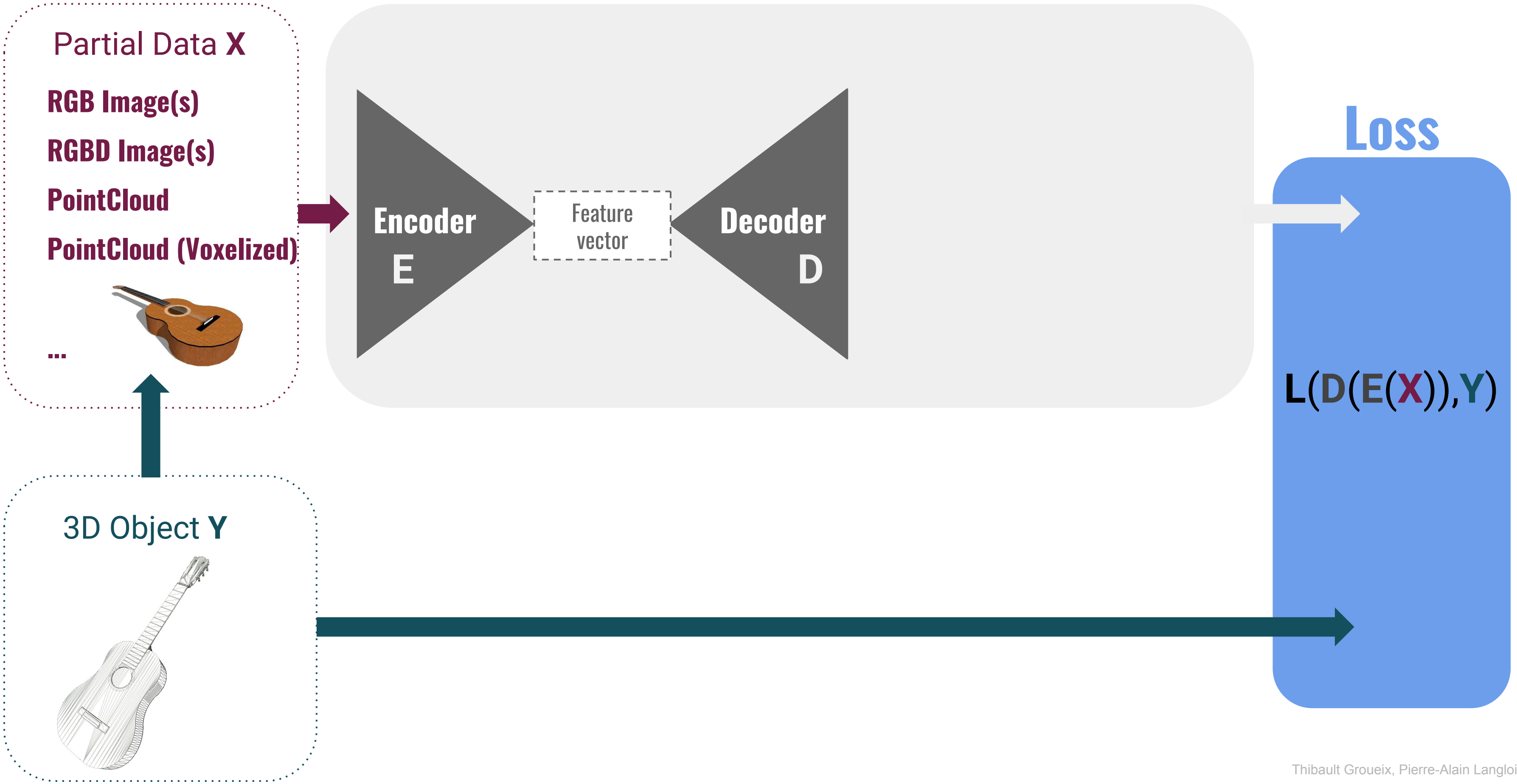
Shape predictor

Loss

L

3D Object **Y**

Training setup for 3D reconstruction



Thibault Groueix, Pierre-Alain Langlois, 2019

Training setup for 3D reconstruction

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

PointCloud (Voxelized)

...

3D Object **Y**

Choice of representation ?

Encoder
E

Feature vector

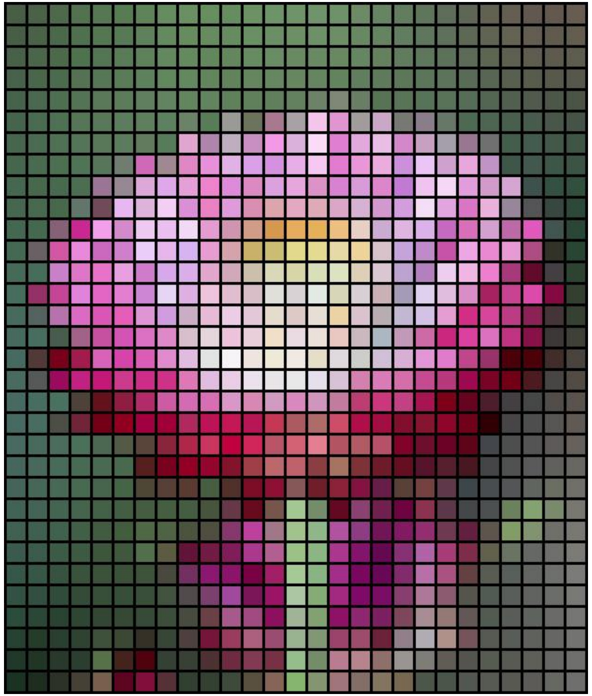
Decoder
D

Loss

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

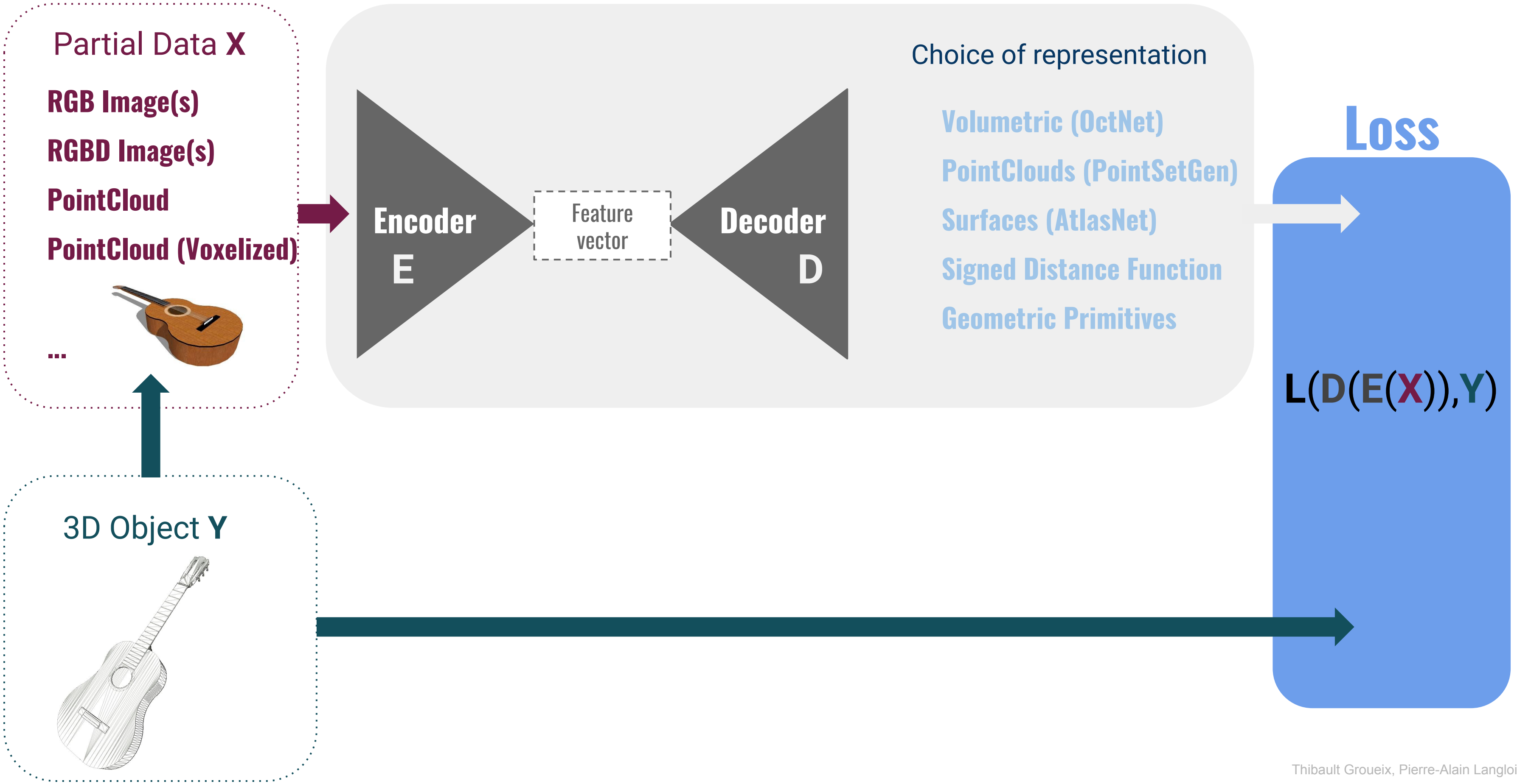
Representations

Obvious in 2D...

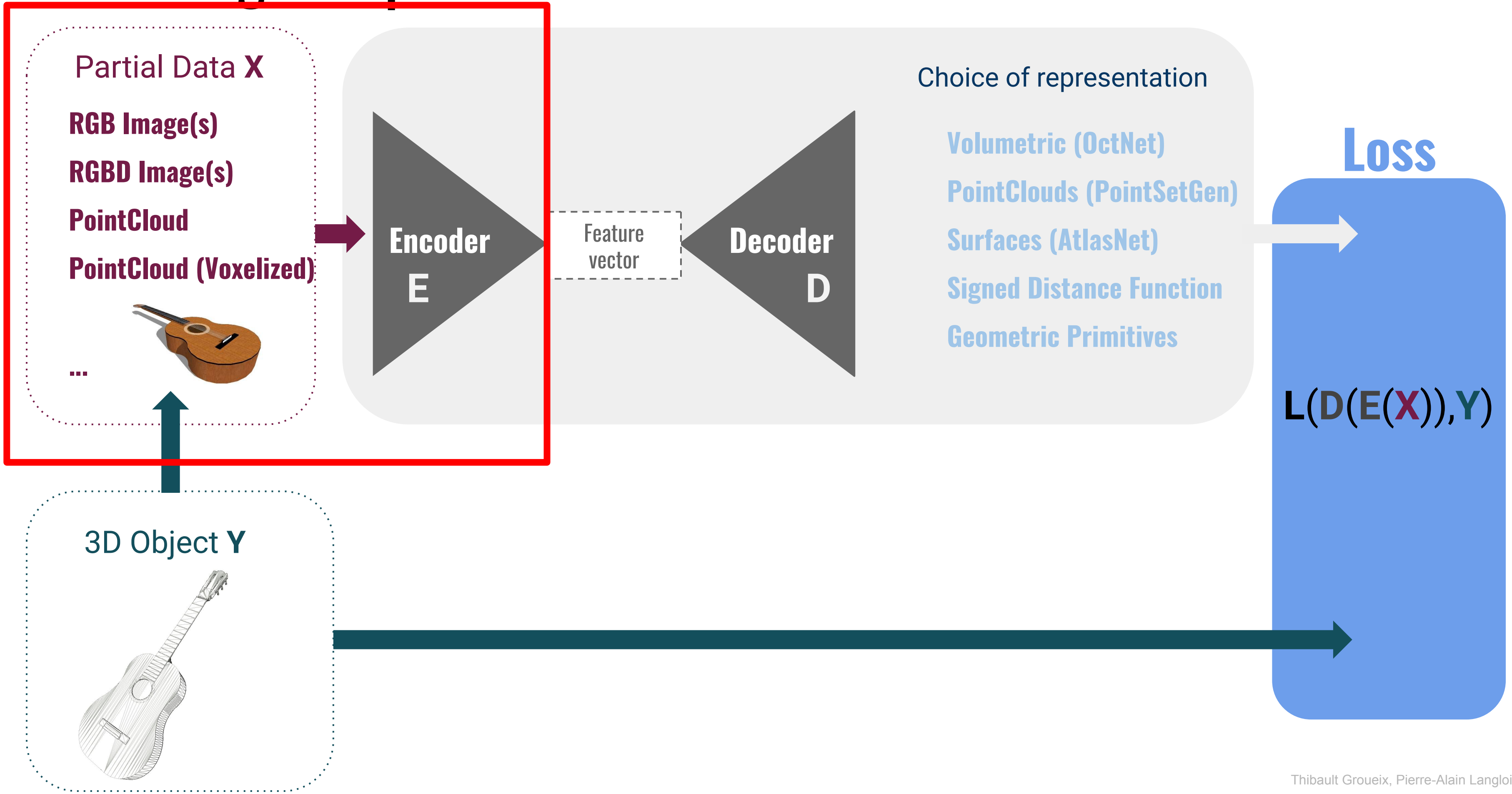


Not so obvious in 3D !

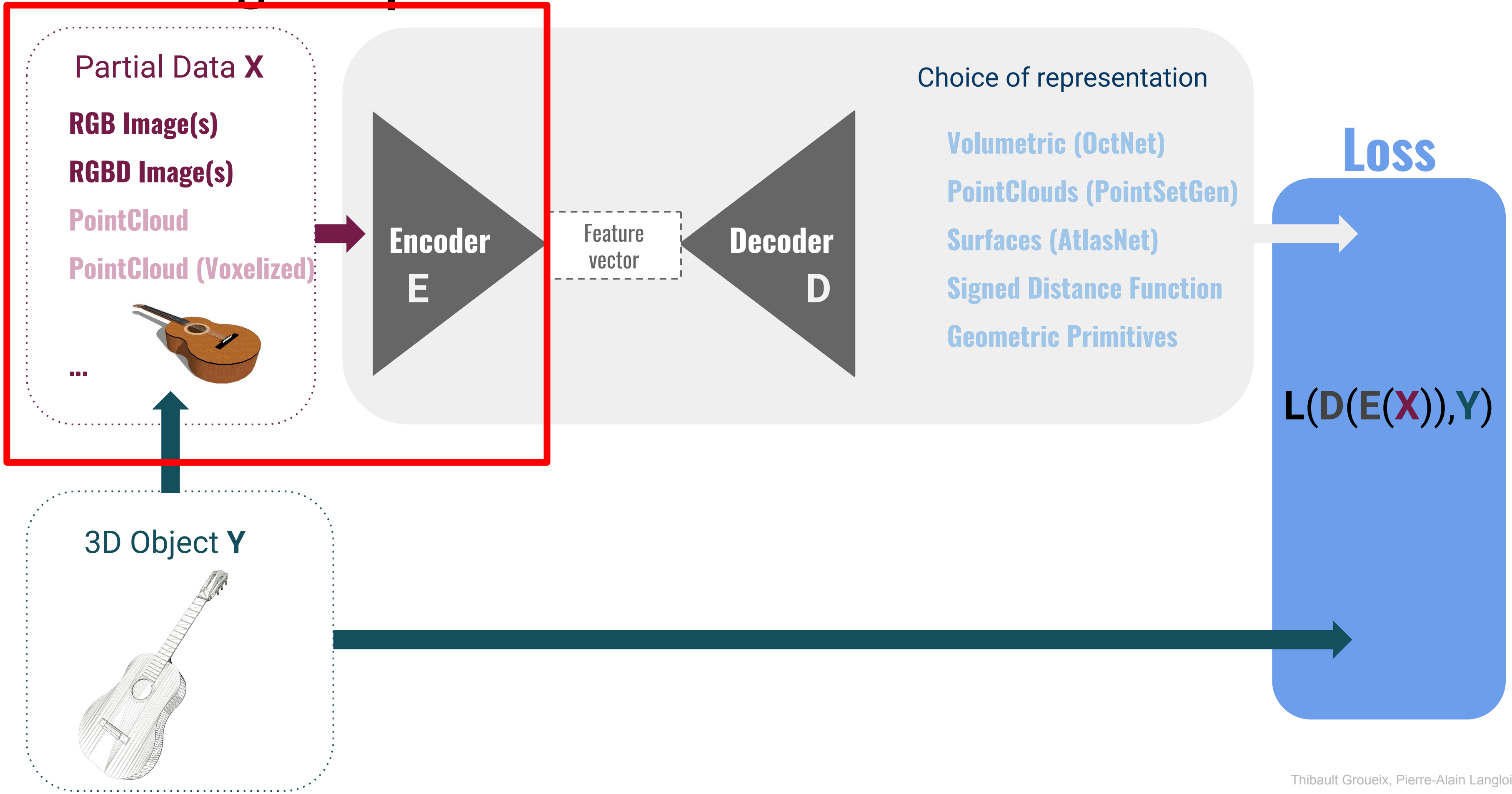
Training setup for 3D reconstruction



Training setup for 3D reconstruction



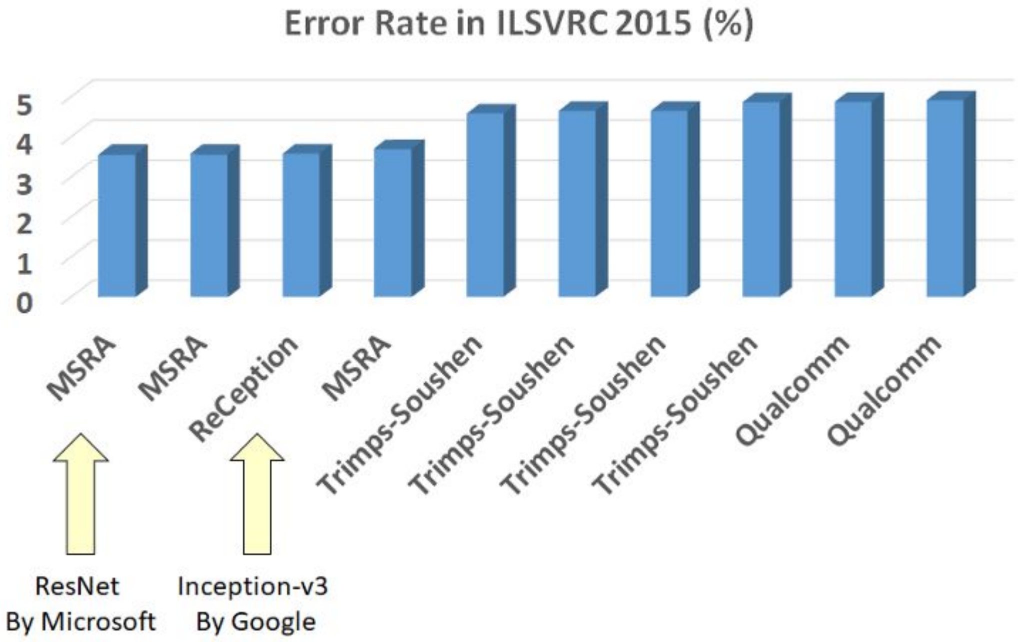
Training setup for 3D reconstruction



Thibault Groueix, Pierre-Alain Langlois, 2019

Encoders for RGB & RGBD images

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

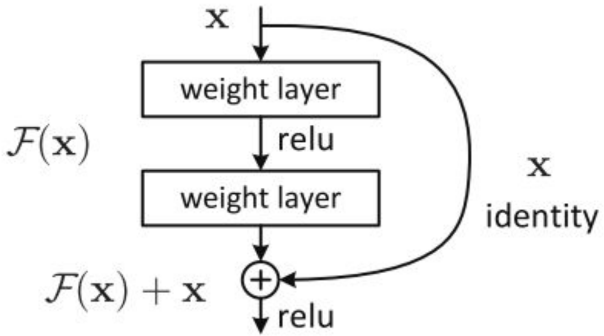


Encoder
E

Encoders for RGB & RGBD images

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

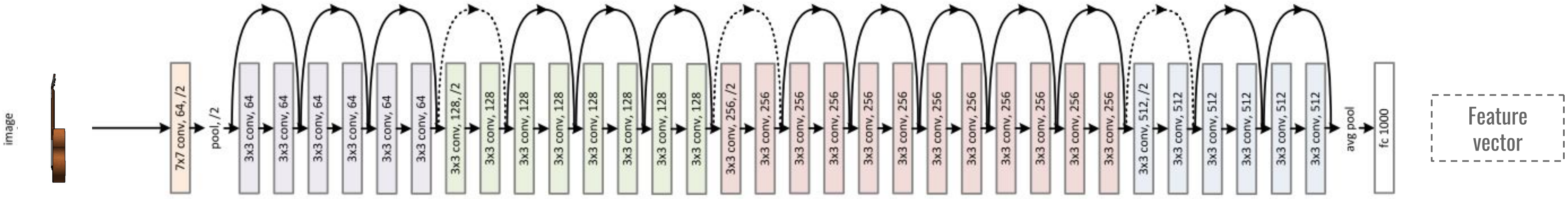
- Resnet [He2015] -> Skip connections
- BatchNorm [Ioffe2015]



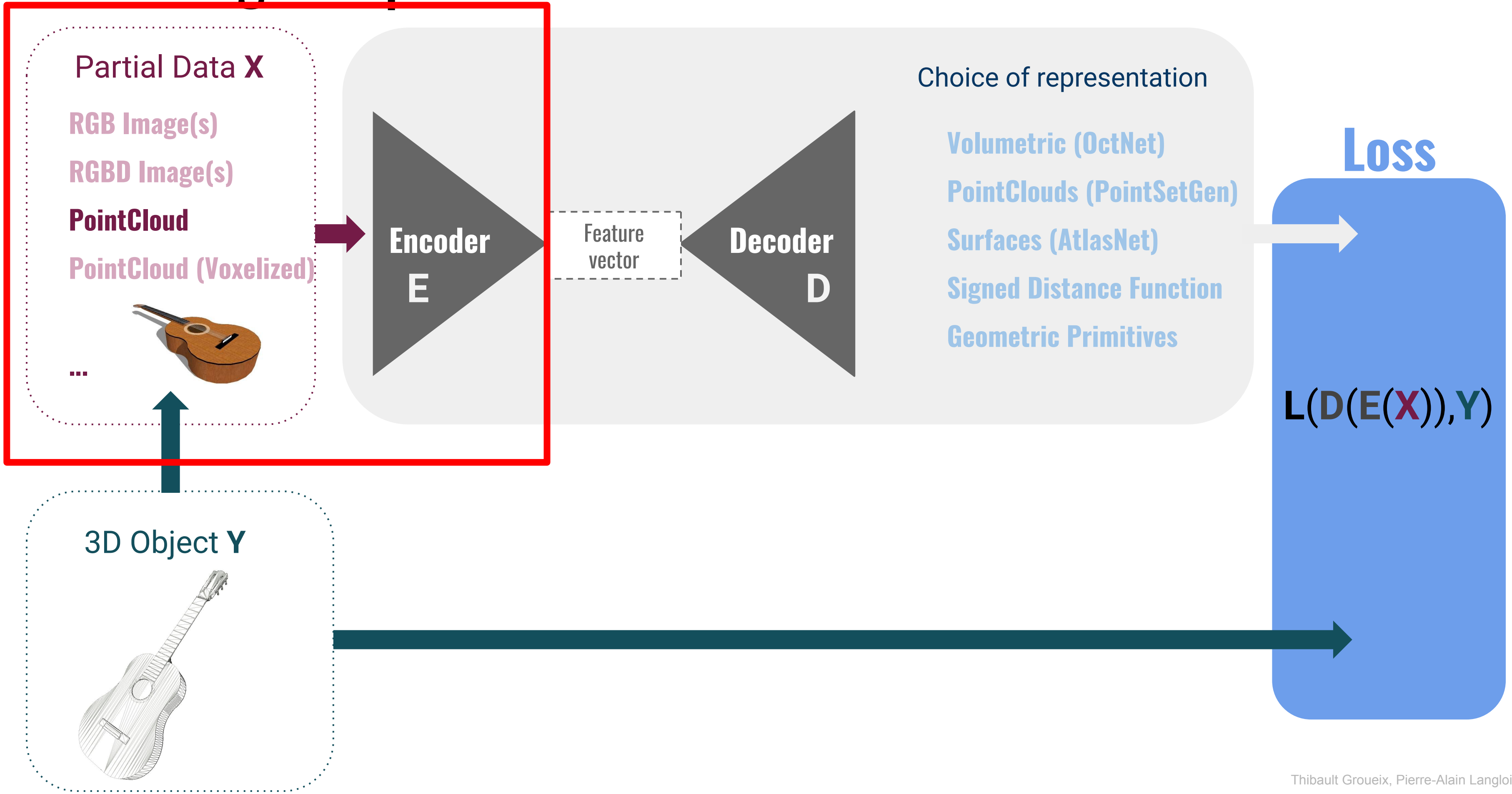
Resnet 34 [He2015]

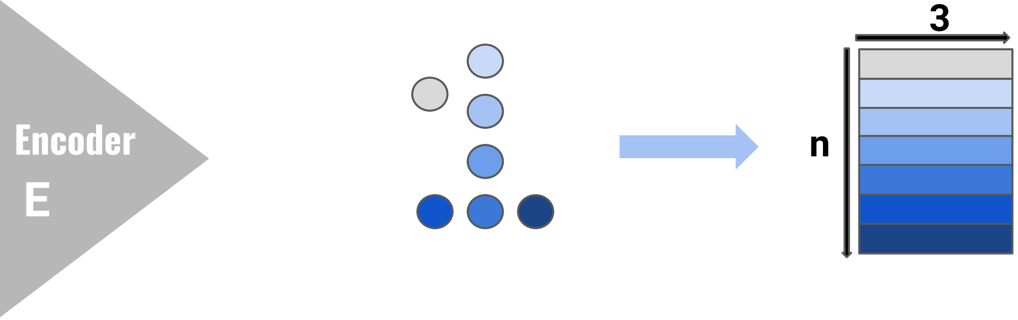
Encoder

E



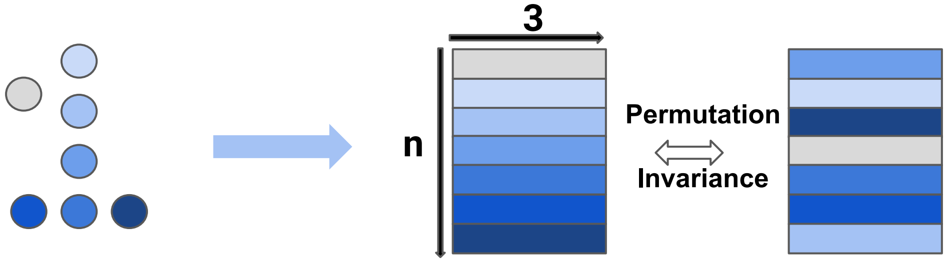
Training setup for 3D reconstruction




Encoder
E
3
n

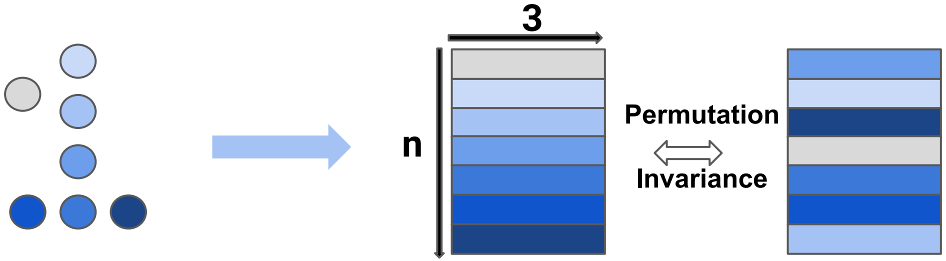
Encoder
E

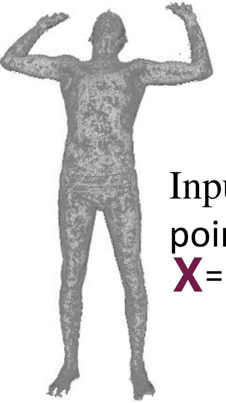
PointNet [Qi2017]



PointNet [Qi2017]

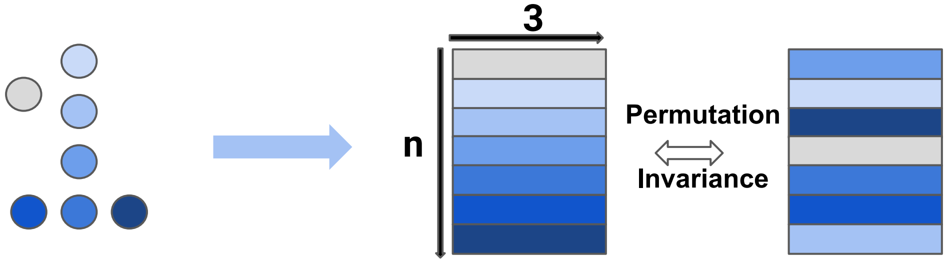
Encoder
E



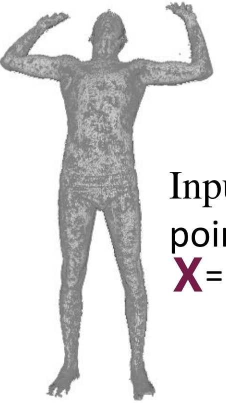


Input
pointcloud
X $= (x_1, x_{2,} \dots, x_n)$

Encoder
E



PointNet [Qi2017]



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

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

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

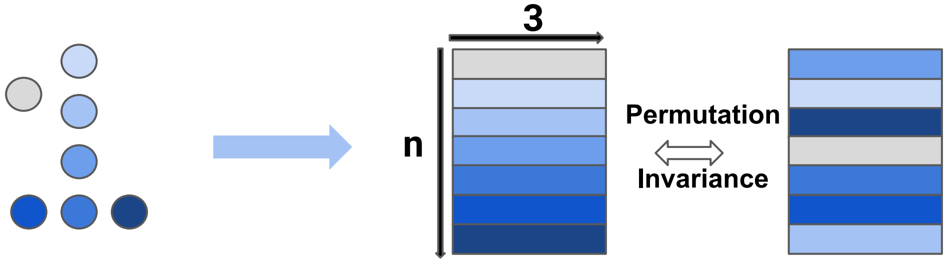
$x_{\ldots} = (2,3,2) \rightarrow$

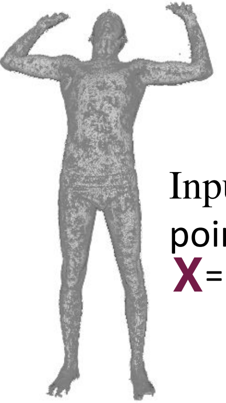
$x_n = (2,3,4) \rightarrow$

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

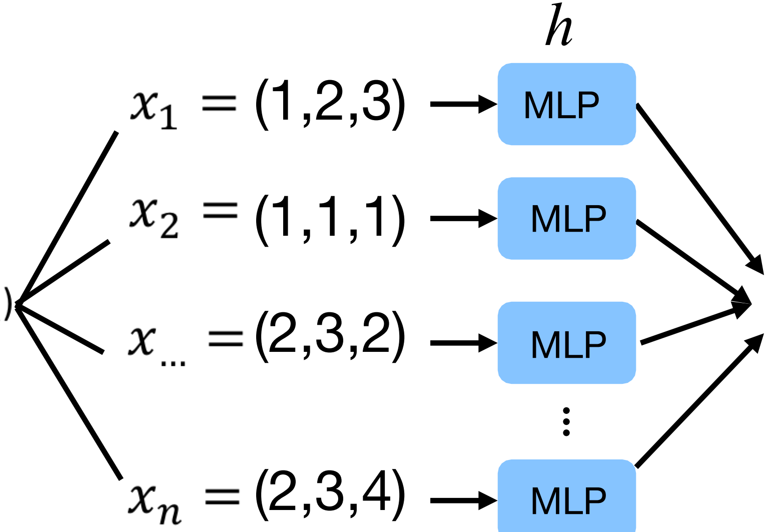
Encoder
E

PointNet [Qi2017]



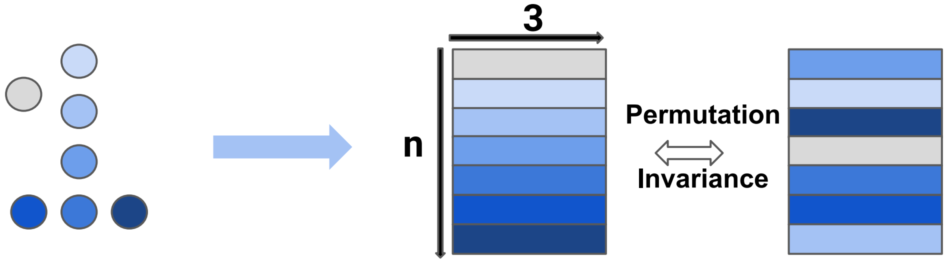


Input
pointcloud
$\mathbf{X} = (x_1, x_2, \ldots, x_n)$

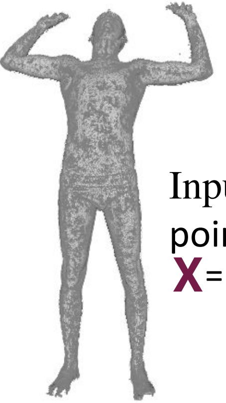


$$\mathbf{E}((x_1, x_2, \ldots, x_n)) = \quad h(x_1), \ldots, h(x_n)$$

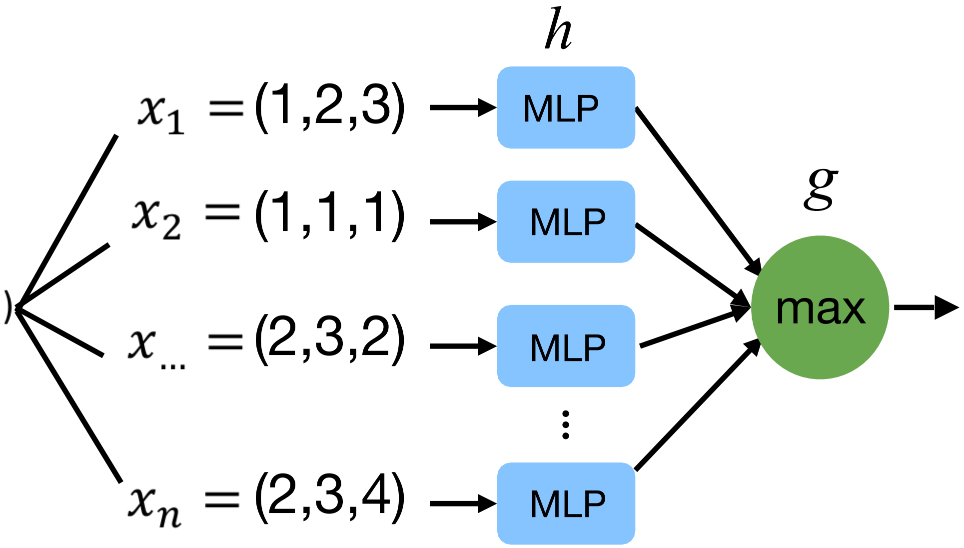
Encoder
E



PointNet [Qi2017]



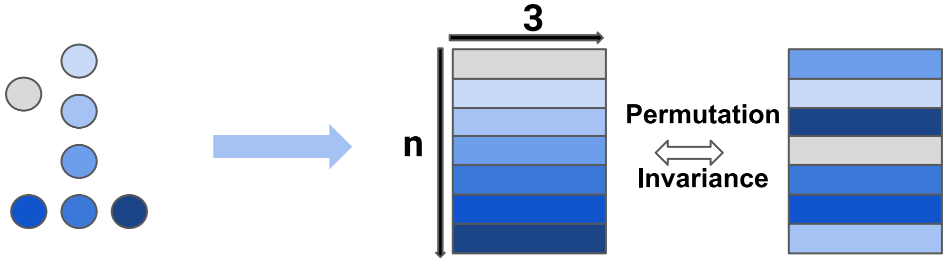
Input
pointcloud
X= $(x_1, x_2, \ldots, x_n)$

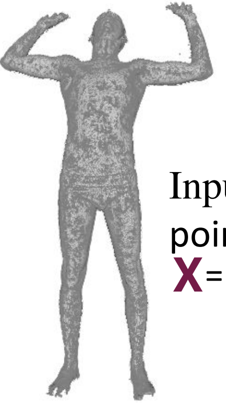


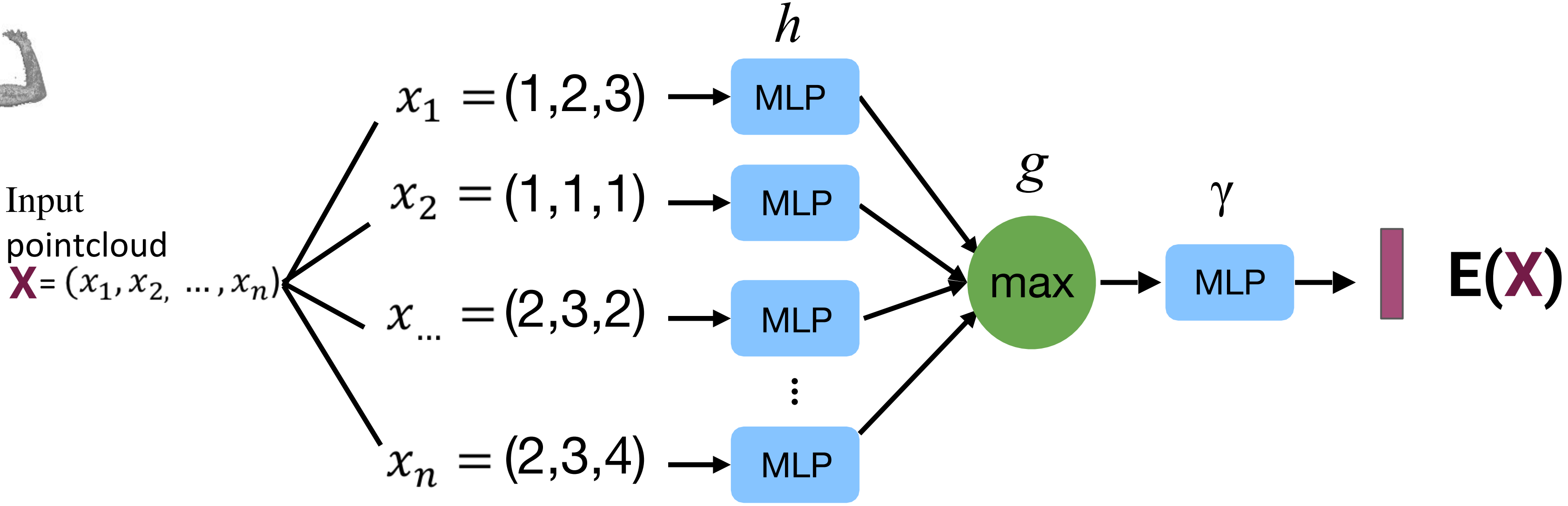
$$\mathbf{E}((x_1, x_2, \ldots, x_n)) = \quad g(\, h(\, x_1\,), \,\ldots,\, h(\, x_n\,)\,)$$

Encoder
E

PointNet [Qi2017]

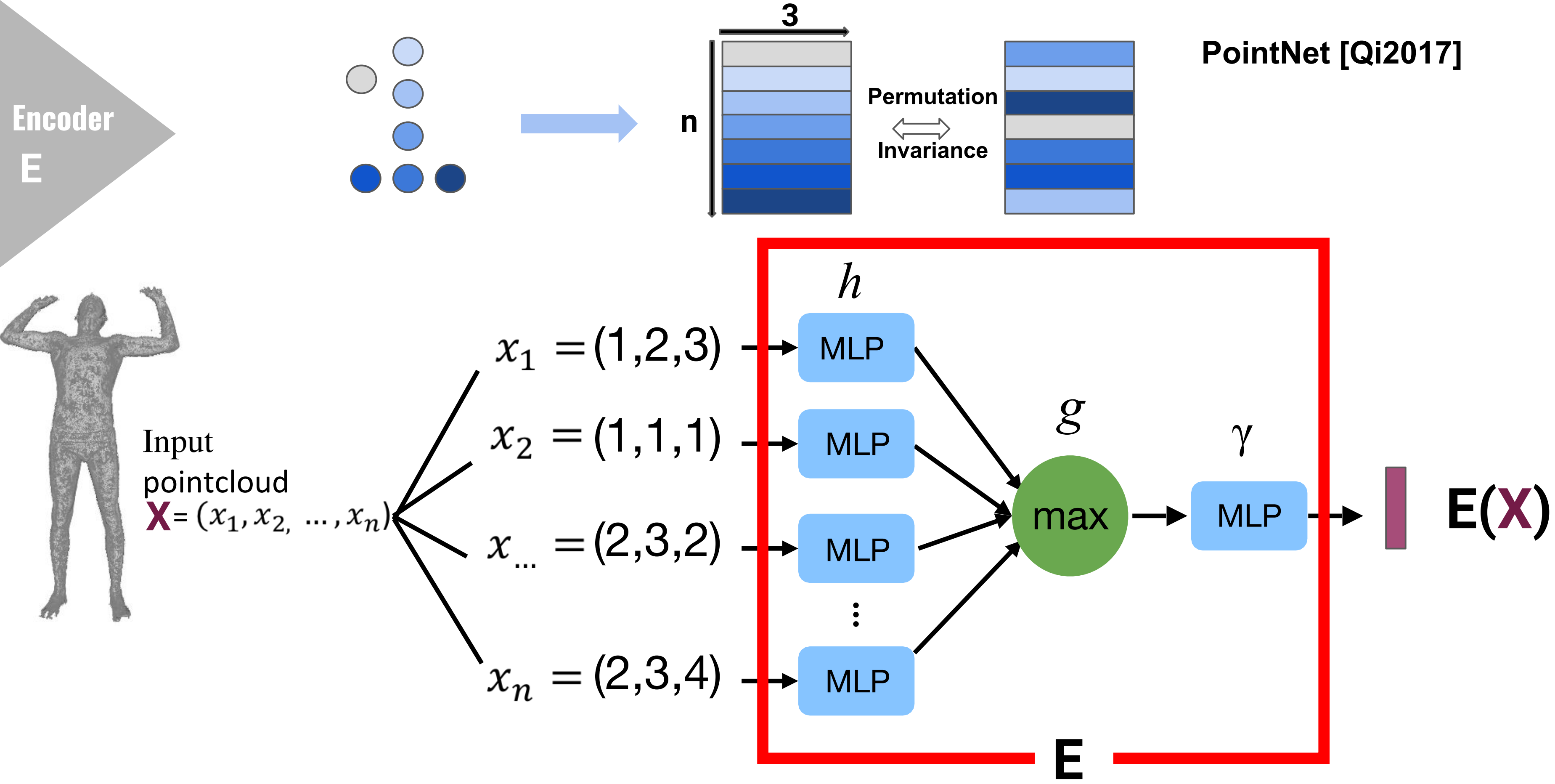






$$\mathbf{E}((x_1, x_2, \ldots, x_n)) = \gamma(\, g(\, h(\, x_1\,), \ldots, \; h(\, x_n\,)\,)\,)$$

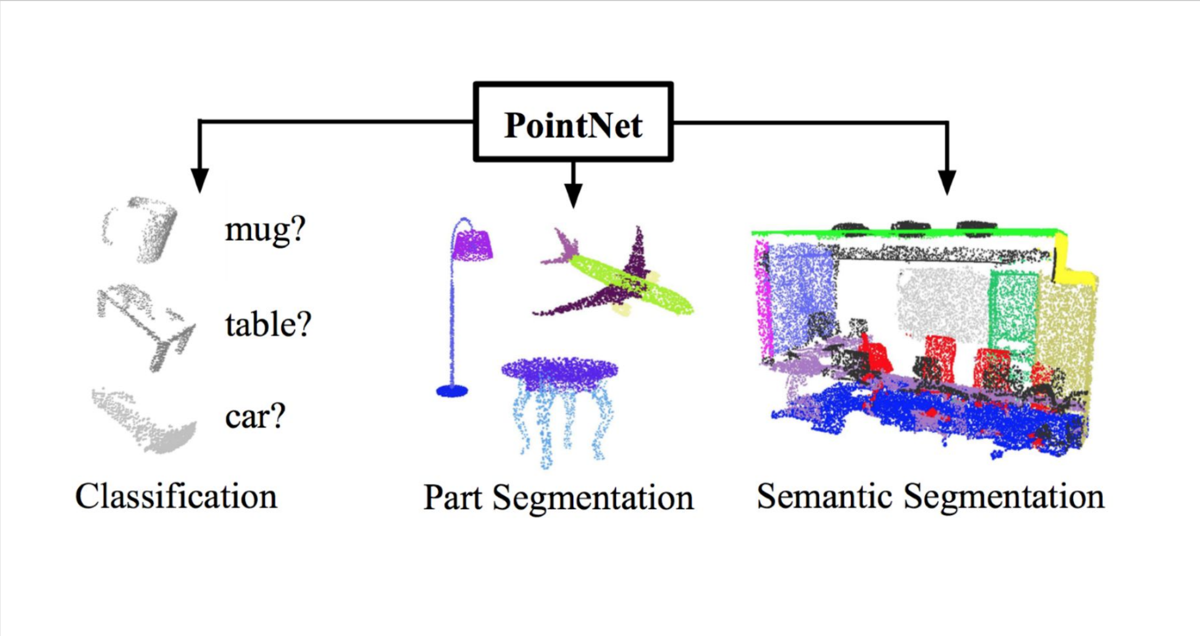
PointNet [Qi2017]



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

Encoder E

Results : Unified framework for various tasks

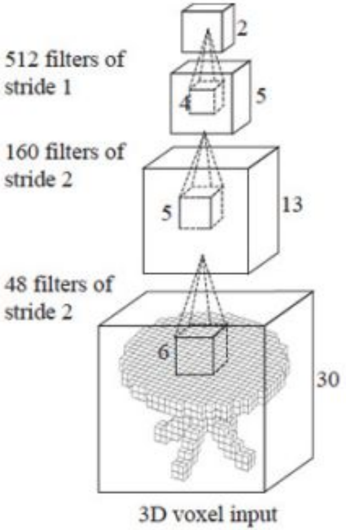


Credit [**Qi2017**]

PointNet Limitations Credit [Qi2017]

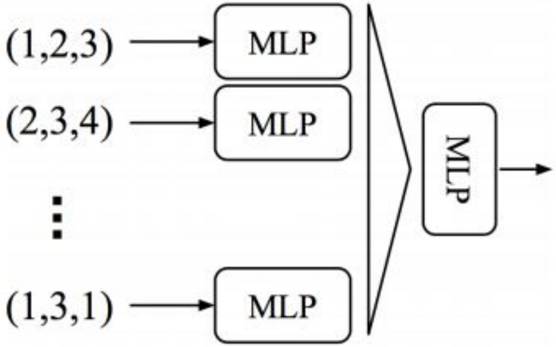
- Hierarchical Feature Learning
- Increasing receptive field

Global Feature Learning
Receptive field:
one point OR all points



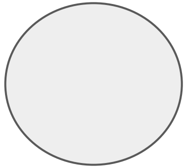
3D CNN (Wu et al.)

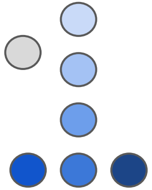
v.s.



PointNet (vanilla) (Qi et al.)

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

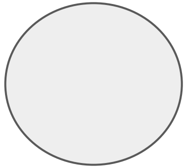
 = PointNet Module



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

Encoder

E



= PointNet Module

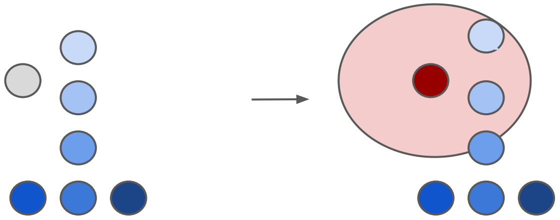
$\mathbf{E(X)}$

Encoder

E

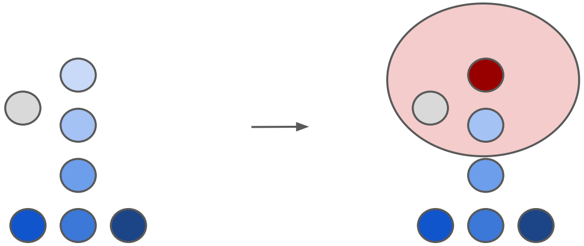
Key idea : Global information is computed in 1 stage : the max function.

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



Encoder E

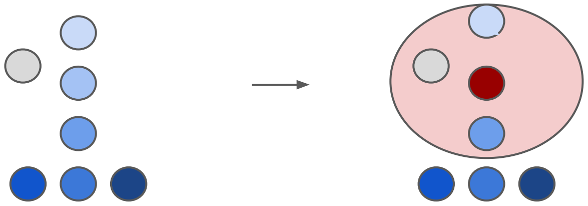
Key idea : Global information is computed in 1 stage : the max function.
Inspired by their success in images, can we build hierarchical filters ?



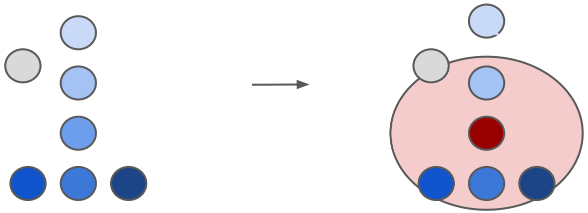
Encoder

E

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

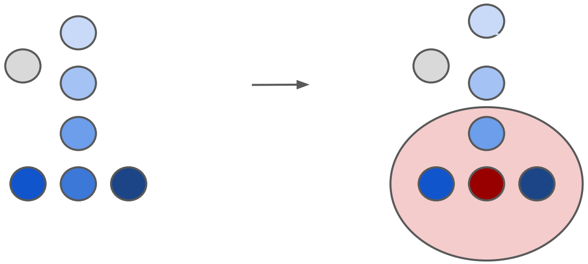


Key idea : Global information is computed in 1 stage : the max function.
Inspired by their success in images, can we build hierarchical filters ?



Encoder
E

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



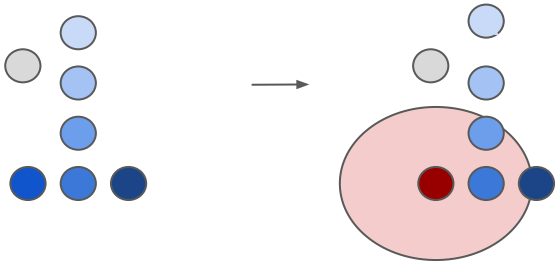
Thibault Groueix, Pierre-Alain Langlois, 2019

Encoder

E

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

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

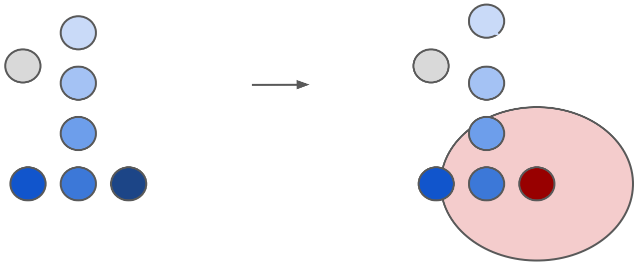


Encoder

E

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

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

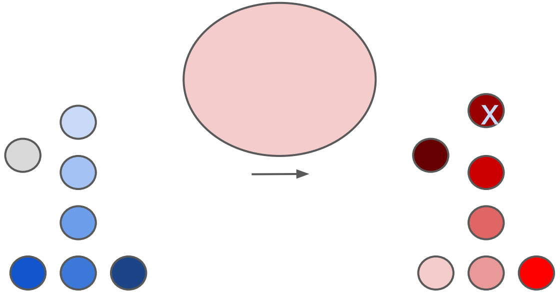


Encoder

E

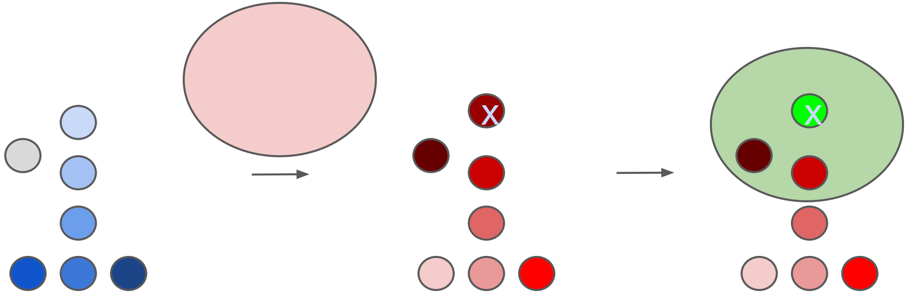
Key idea : Global information is computed in 1 stage : the max function.

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



Encoder
E

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

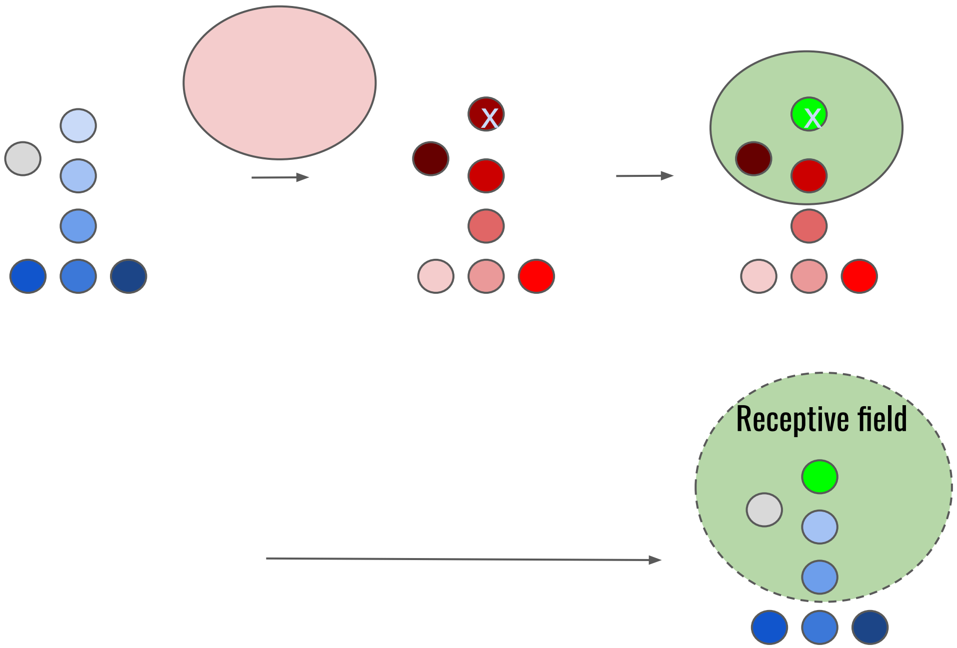


Encoder

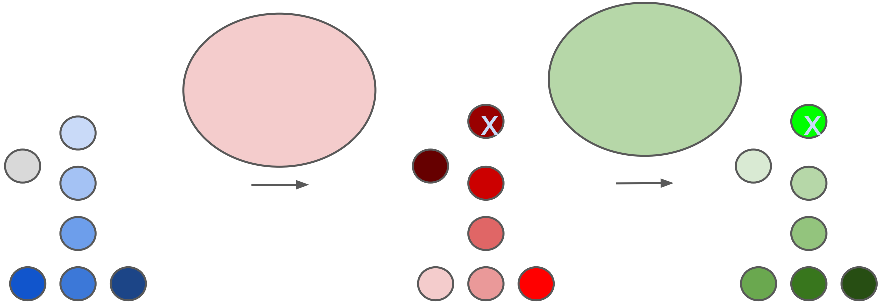
E

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

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



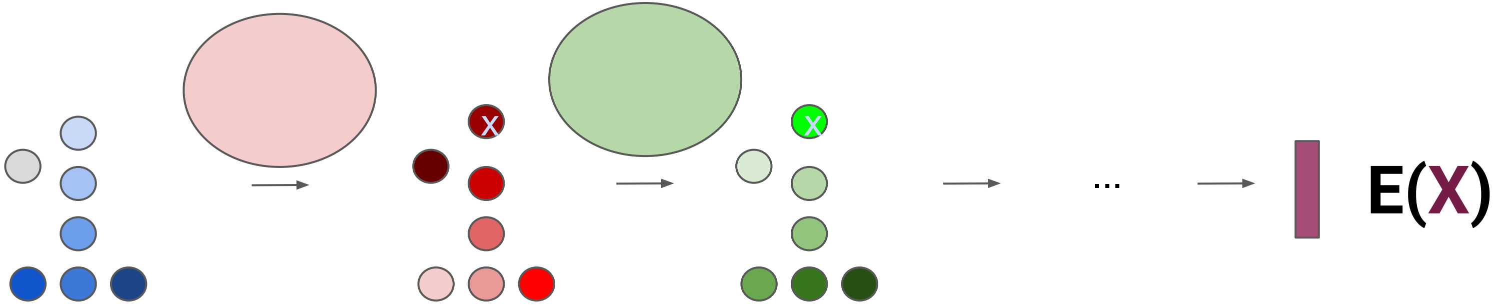
Key idea : Global information is computed in 1 stage : the max function.
Inspired by their success in images, can we build hierarchical filters ?



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

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

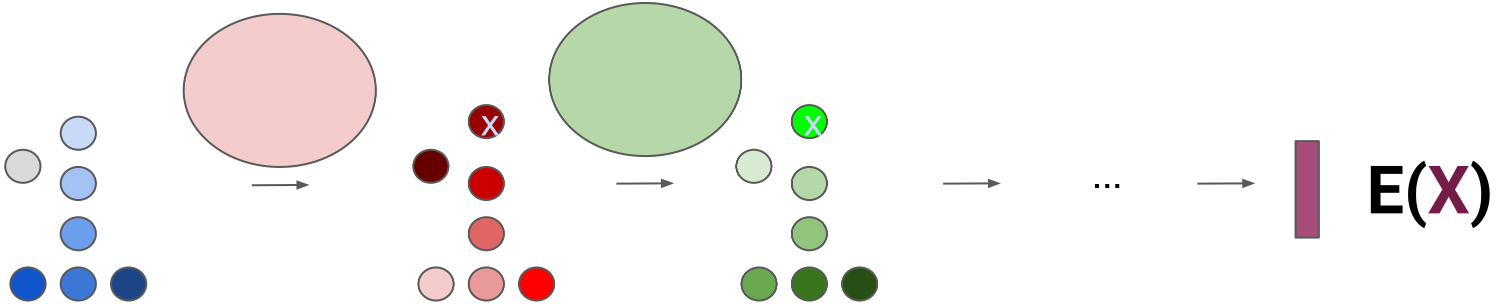
Encoder
E



Encoder

E

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



Key considerations :

❖ Define a receptive field : Ball Query(PointNet++ [Qi2017b, Simonovsky2017]) ? Nearest Neighbors ? Nearest Neighbors in 8 quadrant (pointSIFT [Jiang2018]) ?

❖ Choose a metric : Euclidean ? Geodesic ?

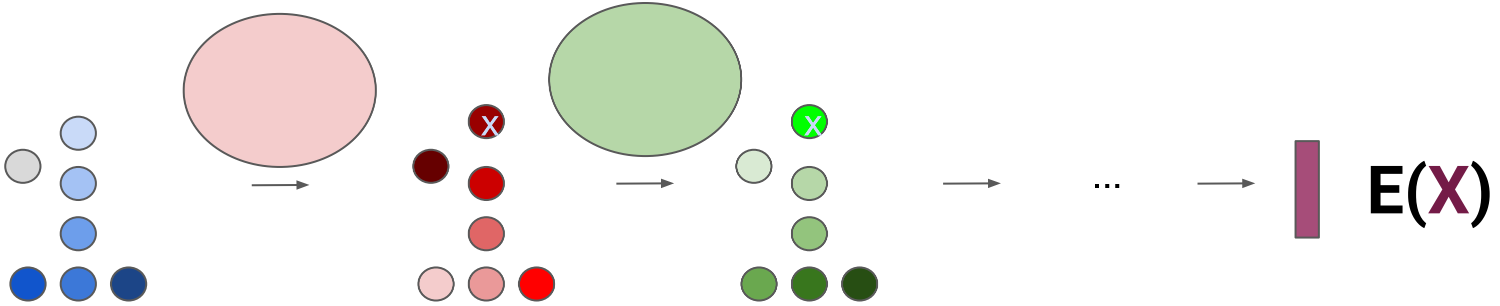
❖ Choose the features : 3D input space features ? Current Layer features (Dynamic Graph CNN [Wang2018]) ?

❖ Global coordinates ? Local coordinates [Qi2017b, Wang2018]?

Encoder

E

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

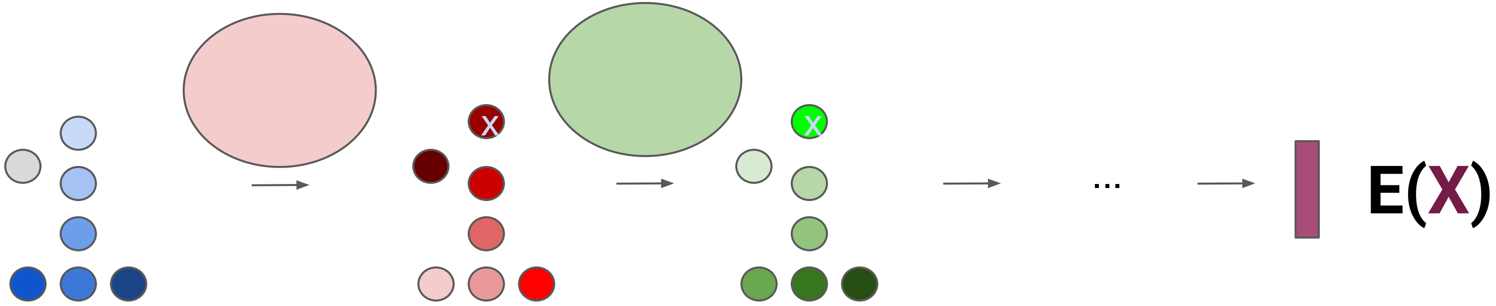


Key considerations :

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

Encoder
E

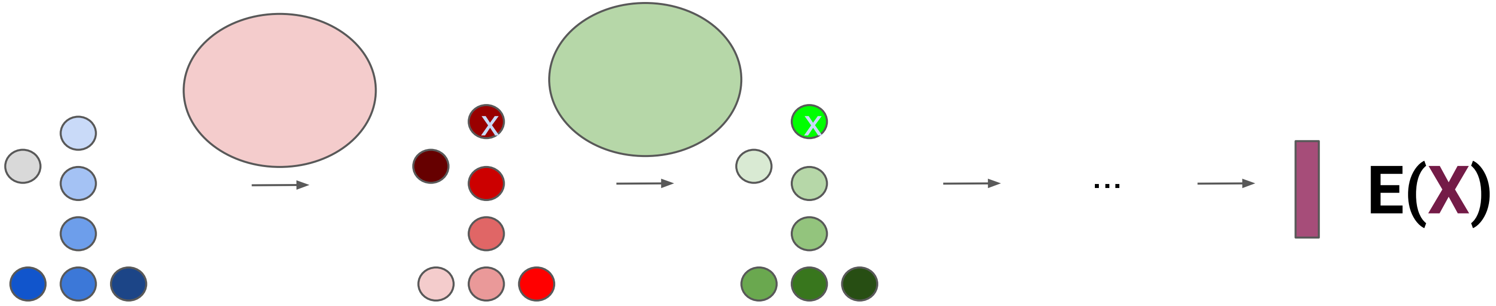
Key idea : Global information is computed in 1 stage : the max function. Inspired by their success in images, can we build hierarchical filters ?



Key considerations :

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

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



Key considerations :

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

Thibault Groueix, Pierre-Alain Langlois, 2019

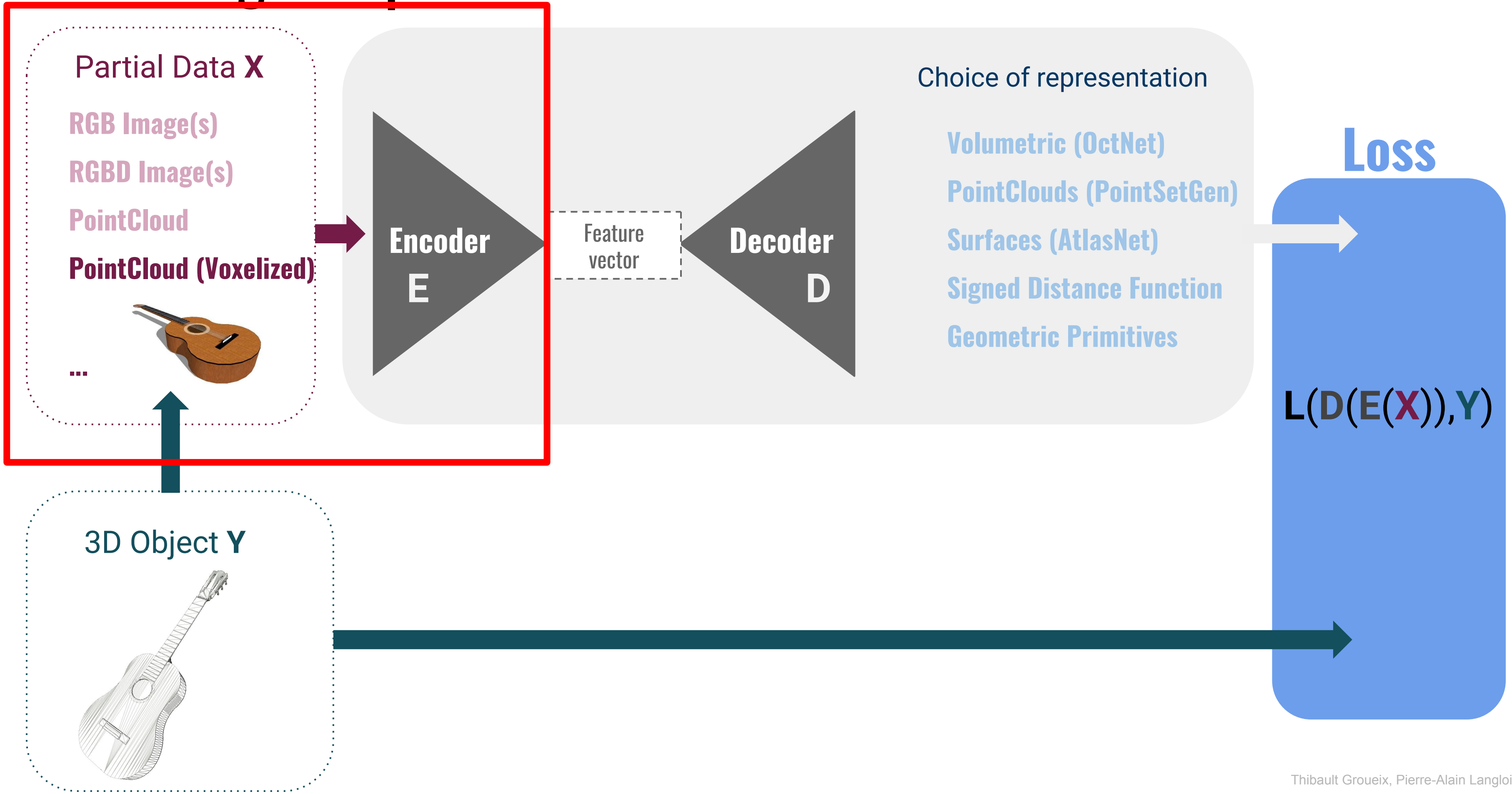
A number of (good) alternatives exists

Encoder

E

- ➔ 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



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

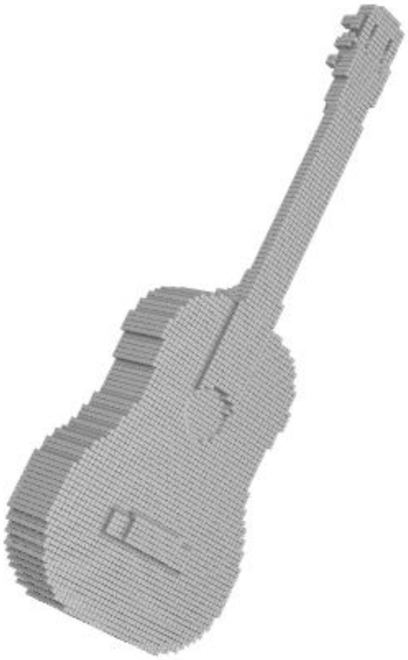
Encoder
E

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

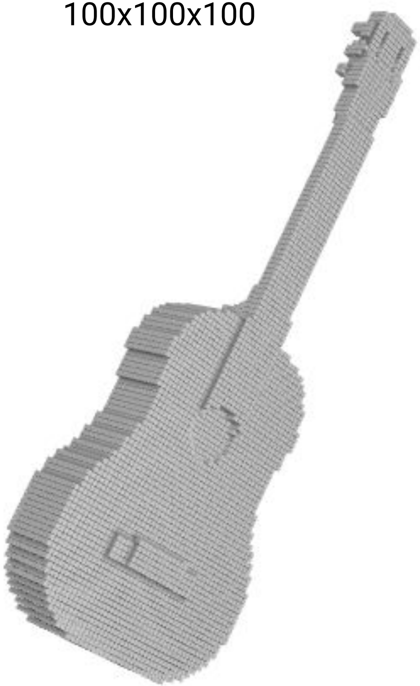


Volumetric representations

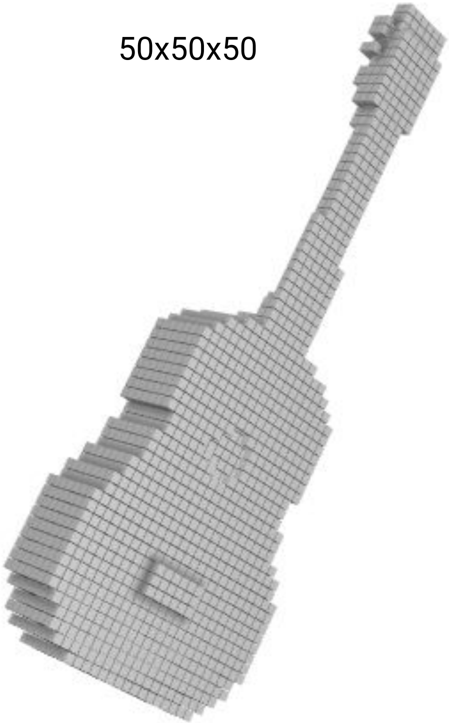
Encoder
E

Decoder
D

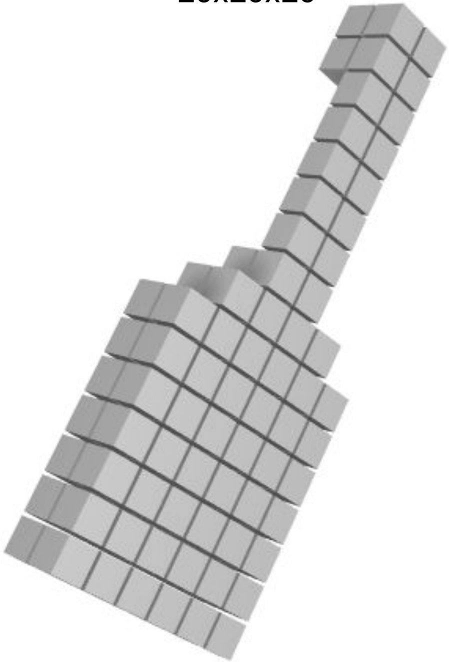
100x100x100



50x50x50



20x20x20



Hybrid Grid-Octree Data Structure

Encoder
E

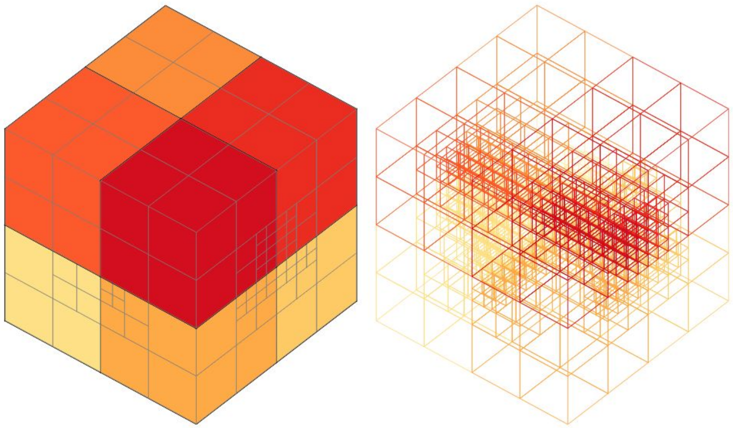
Decoder
D

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

➔ Grid of octrees with fixed small depth : typically 3

➔ Computationally more effective

➔ Good compression rate



OctNet input

- ➔ If a cell contains data from the mesh, it takes value 1 and it is subdivided
- ➔ Otherwise, it takes the value 0
- ➔ Easy to compare with the L2 distance over voxels

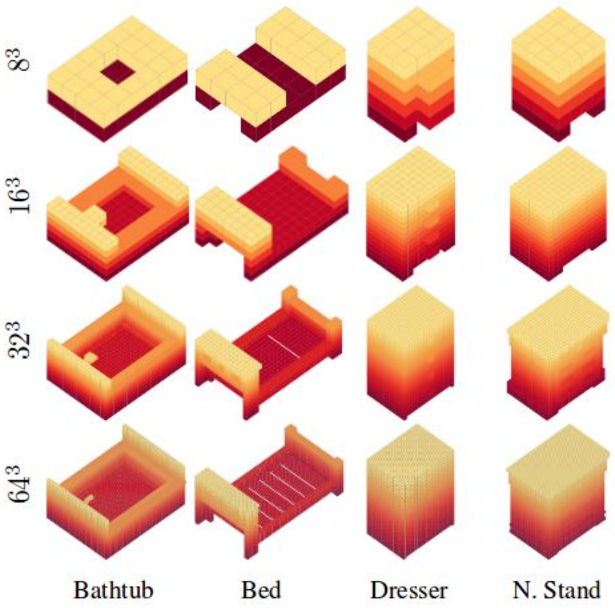


Figure 8: **Voxelized 3D Shapes from ModelNet10.**

Convolutions on Grid-Octree Data Structure

- ➔ Improvement : Inside a given cell the convolution result is the same. We can compute it once.
- ➔ Convolution is computed on the boundaries

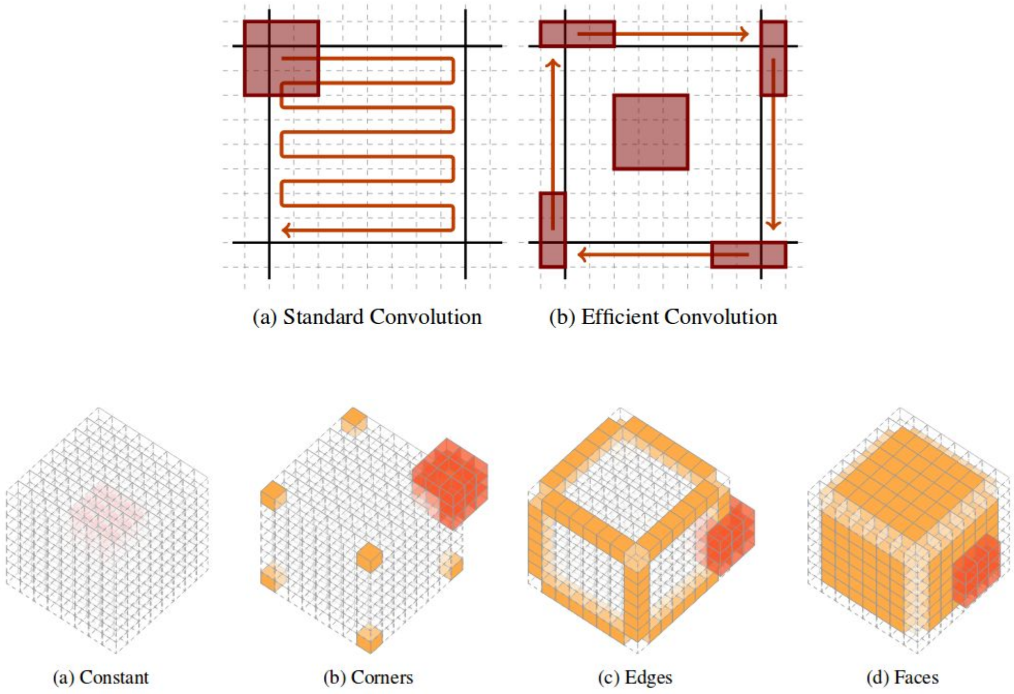


Figure 14: **Efficient Convolution.**

Pooling on Grid-Octree Data Structure

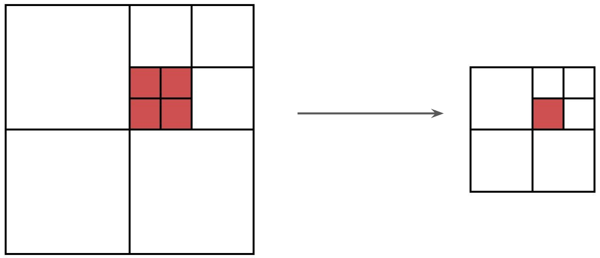
Encoder

E

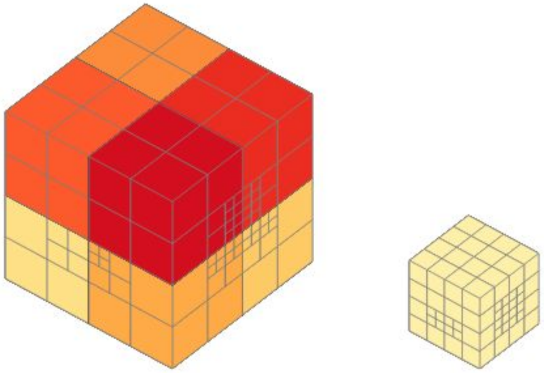
➔ Voxels at maximum resolution are pulled

➔ Voxels at higher resolutions are halved in size

2D example

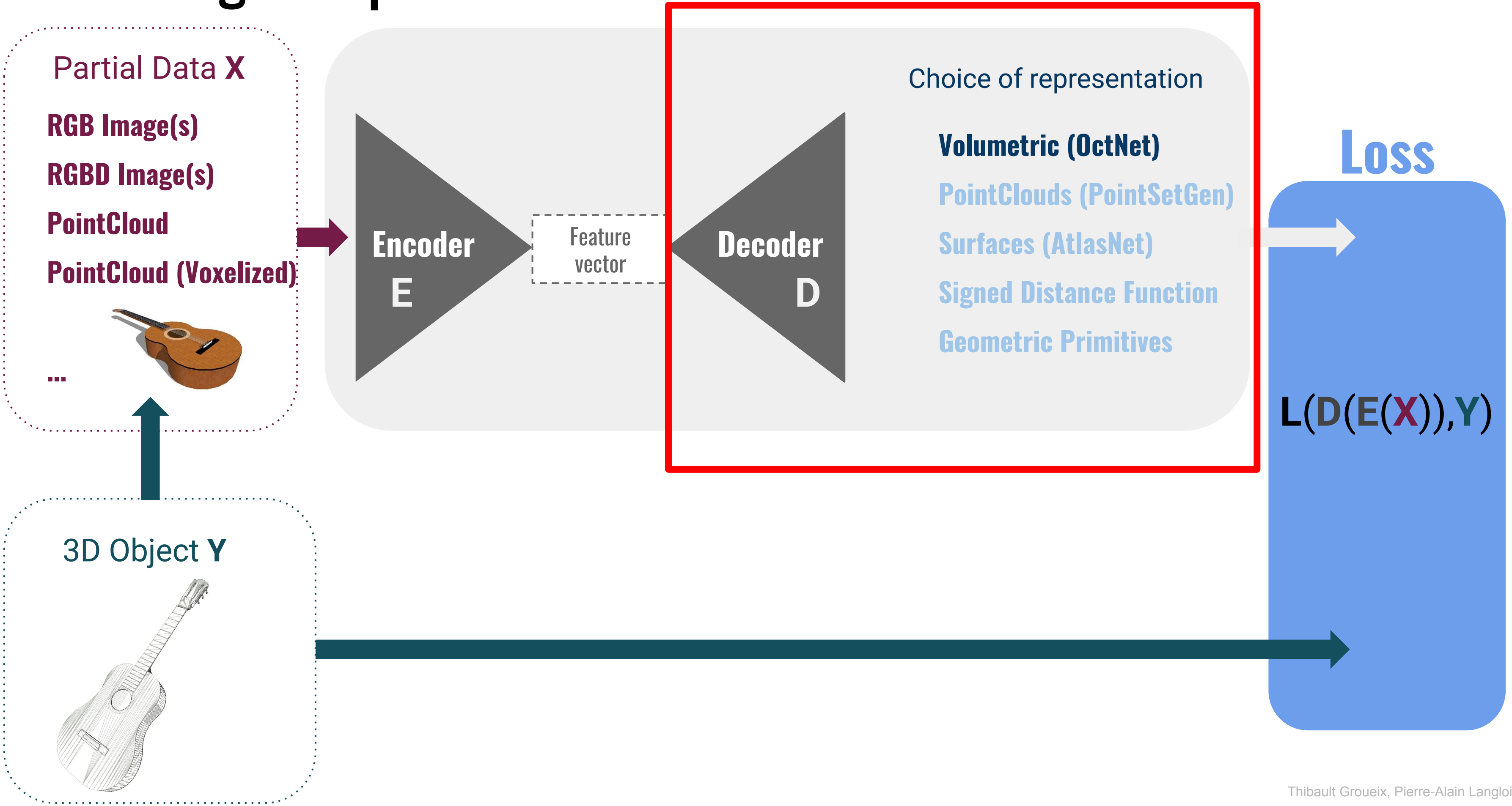


3D



(a) Input (b) Output

Training setup for 3D reconstruction



Decoding towards an octree

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

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

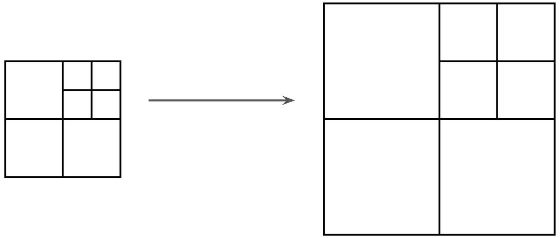
➔ We need to predict the octree structure

Unpooling on Grid-Octree Data Structure

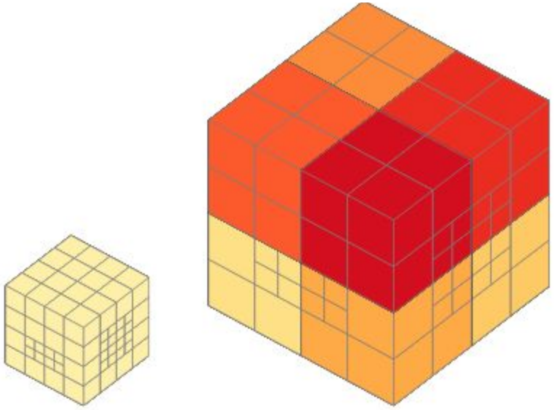
Decoder
D

➔ All nodes double their sizes

2D
example



3D



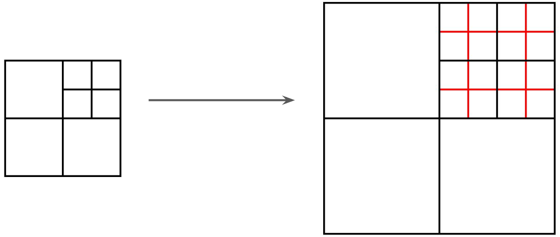
(a) Input (b) Output

Unpooling on Grid-Octree Data Structure

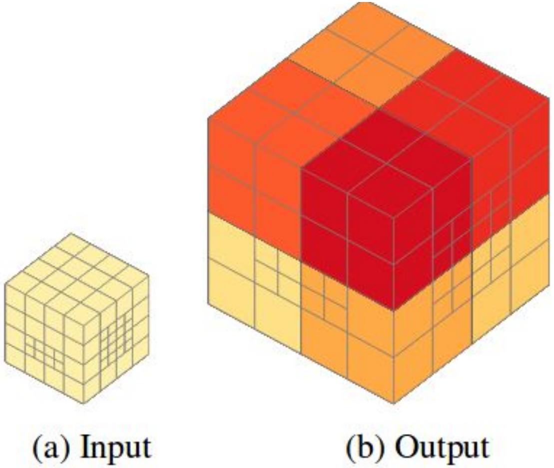
- ➔ All nodes double their sizes

What about capturing details at finer resolution ?

2D example



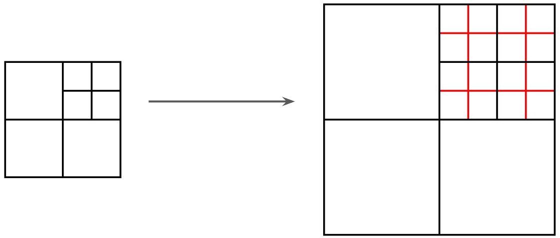
3D



Unpooling on Grid-Octree Data Structure

➔ All nodes double their sizes

2D example

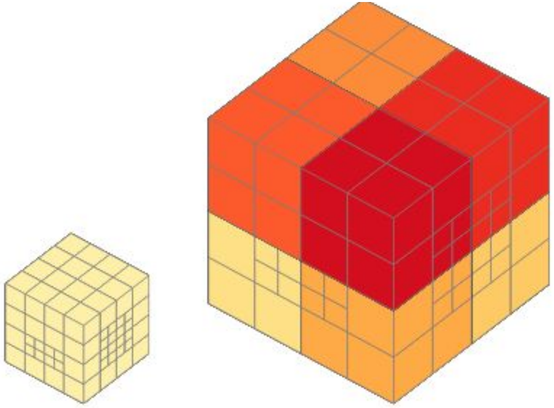


What about capturing details at finer resolution ?

➔ 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]**

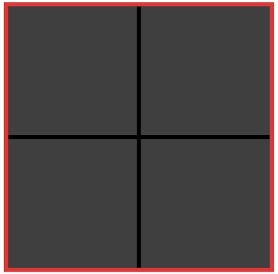
3D



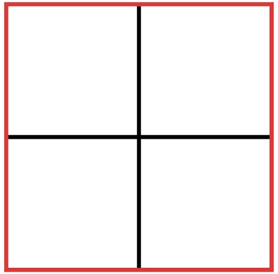
(a) Input (b) Output

Octree generating networks - results

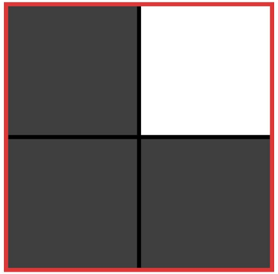
Subdivision is predicted as a classification task. **[Tatarchenko2017]**



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]**

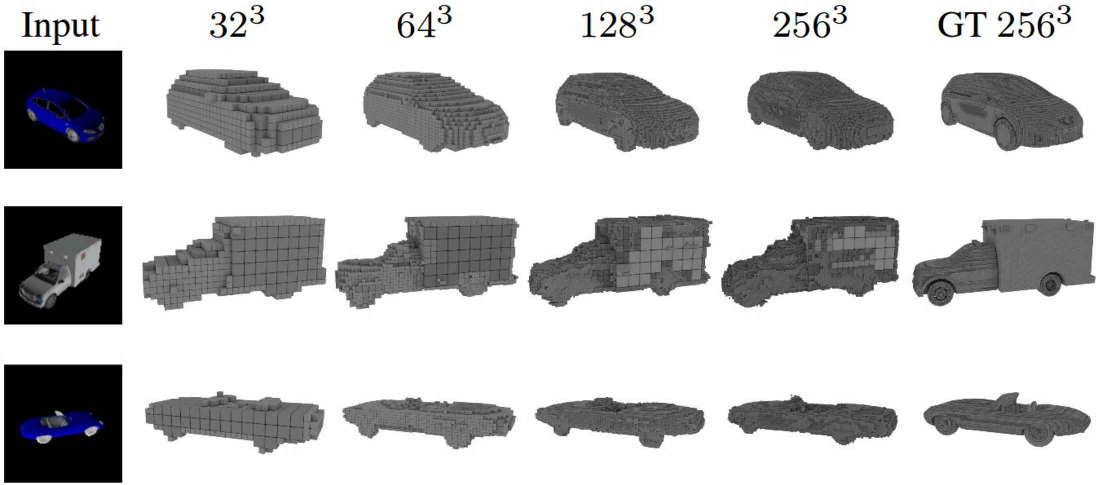


Figure 8. Single-image 3D reconstruction on the ShapeNet-cars dataset using OGN in different resolutions.

Encoder
E

Octree-based reconstruction

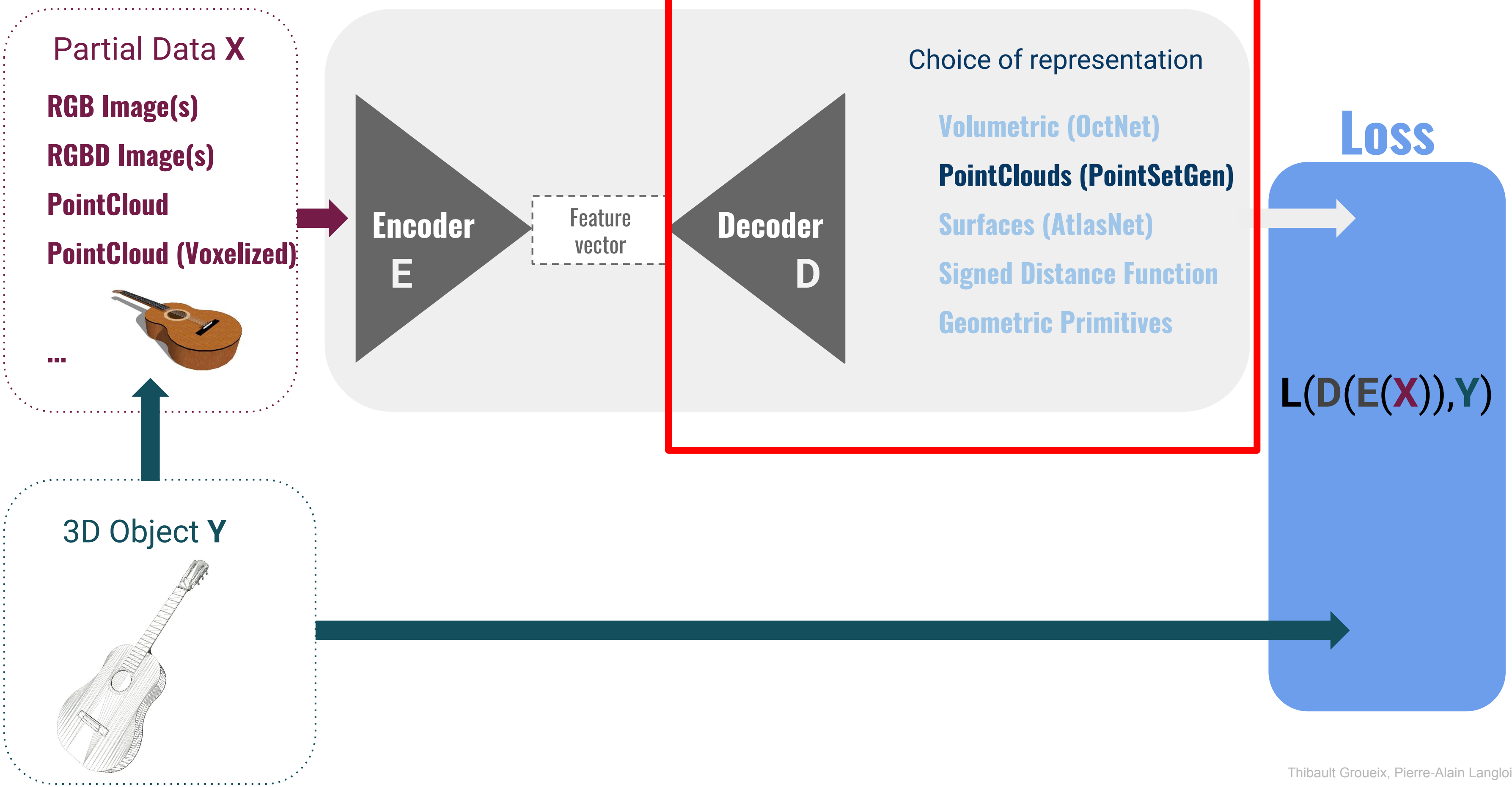
Decoder
D

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

➔ Important improvement in the fight against the curse of dimensionality

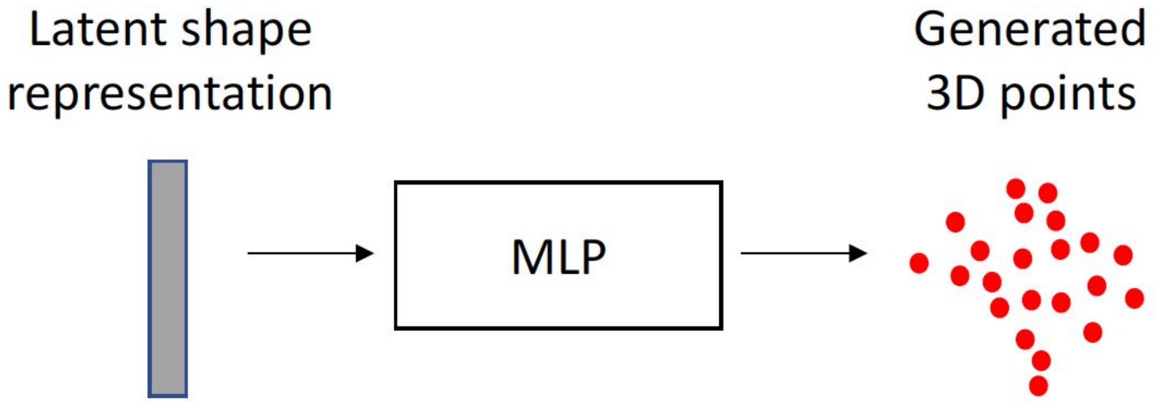
➔ Gives quantitative results regarding the **need for higher resolutions**

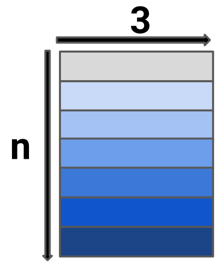
Training setup for 3D reconstruction



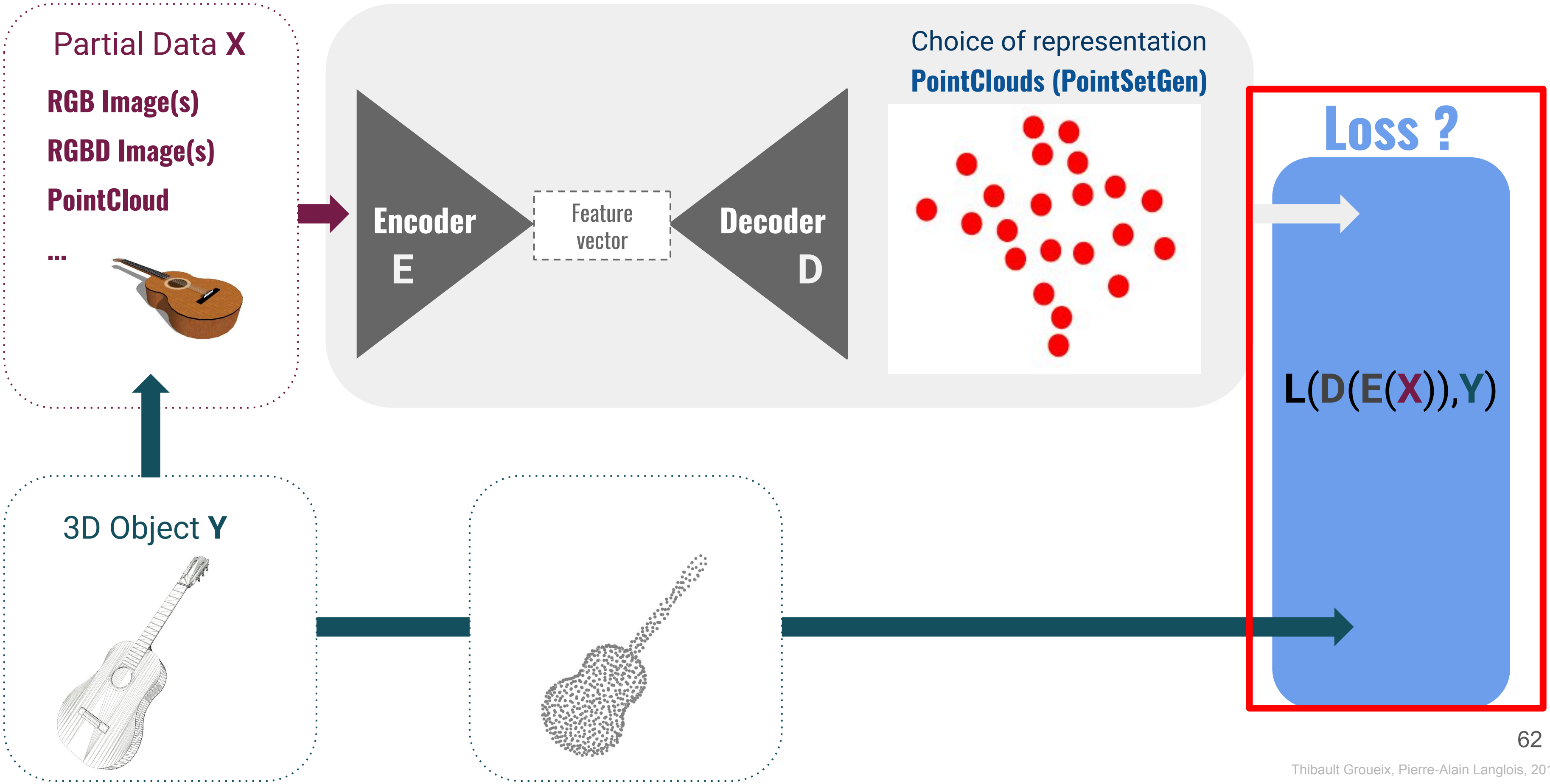
Generating points PointSetGen[**Fan2017**]

Decoder
D

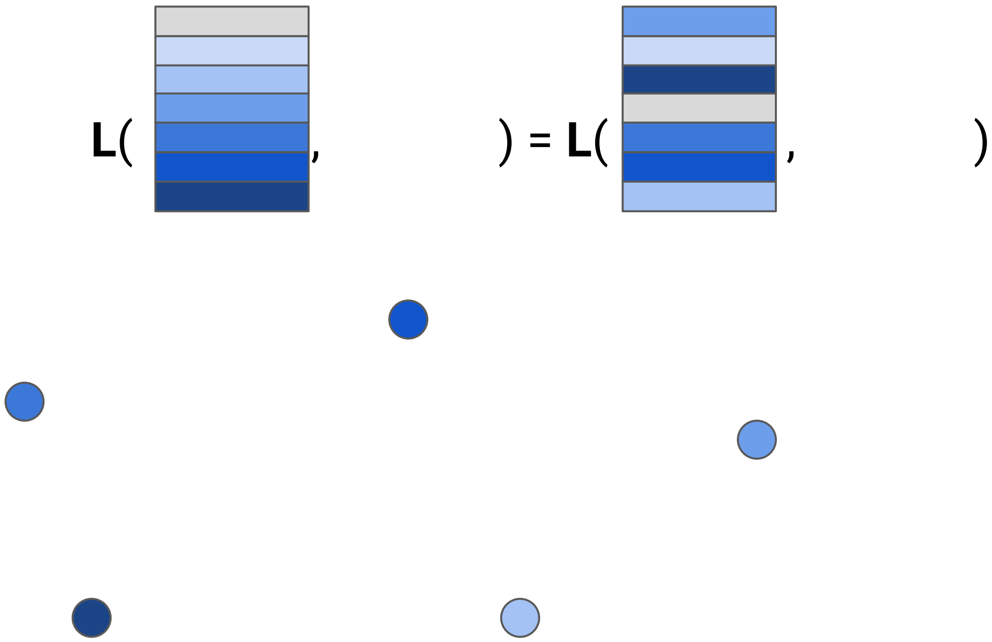




Training setup for 3D reconstruction

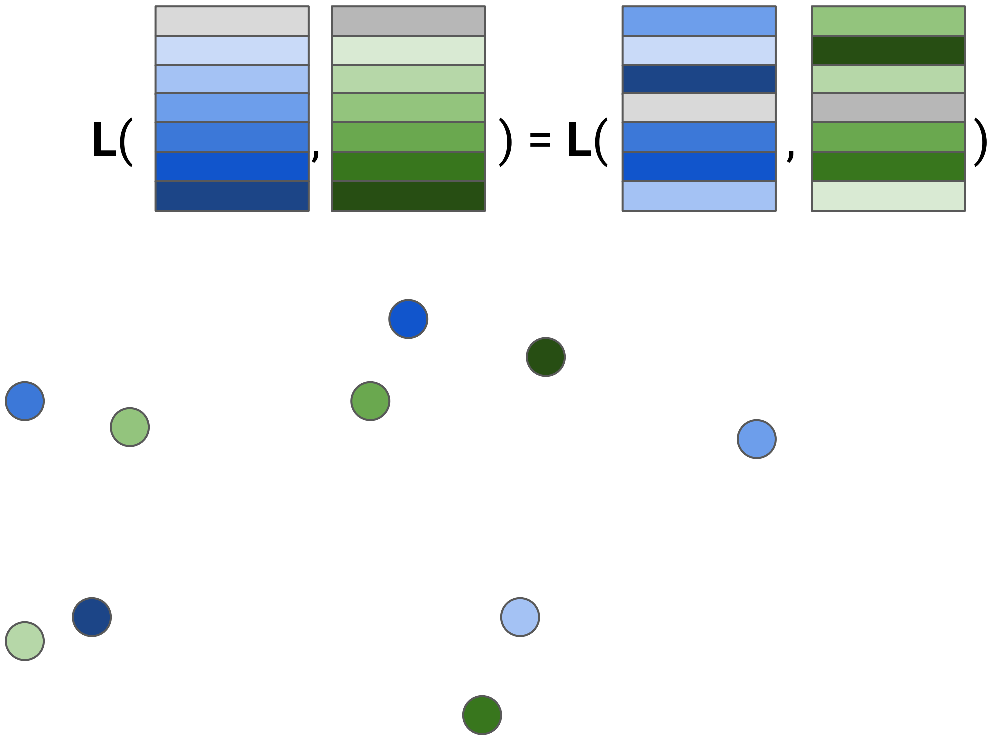


Loss on pointclouds



	Complexity
EMD	n^3
Chamfer	n^2

Loss on pointclouds



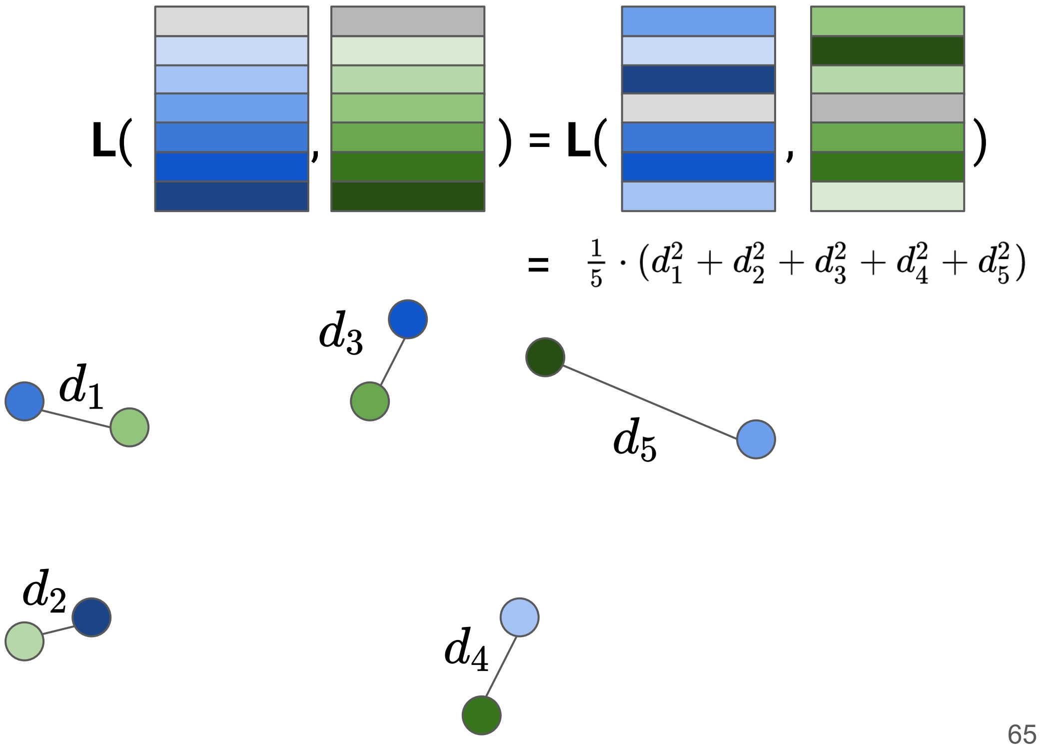
	Complexity
EMD	n^3
Chamfer	n^2

Loss on pointclouds

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

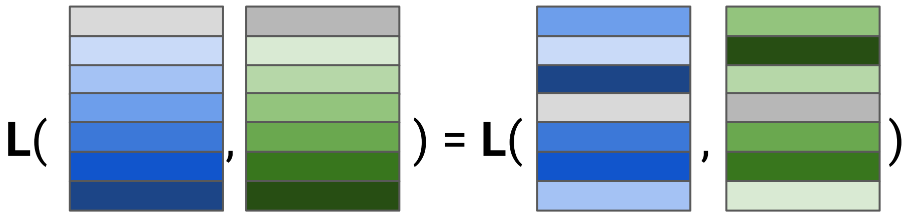
- ➔ Hungarian Algorithm **[Kuhn1955]** ~O(n^3)
- ➔ Simplex based solver through LP formulation ~O(Hungarian)
- ➔ Sinkhorn regularization [**Cuturi2013**] in near linear time [**Altschuler2017**]
- ➔ (1+ε) approximation [**Bertsekas1988**] in ~O(n^3)

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

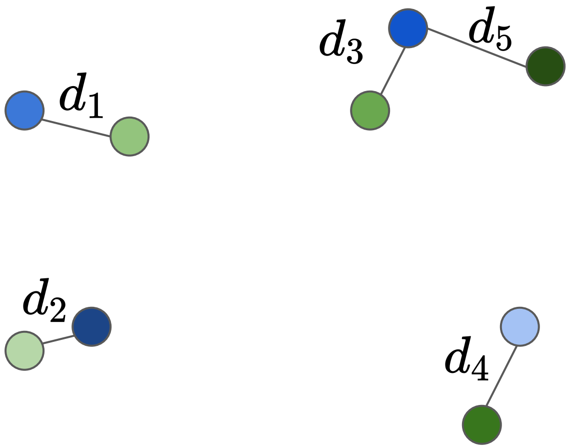


Loss on pointclouds

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



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

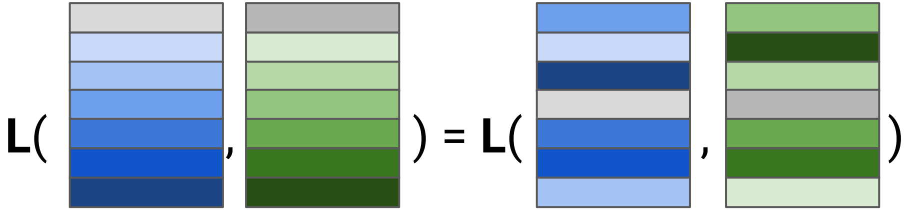


	Complexity
EMD	**n^3**
Chamfer	**n^2**

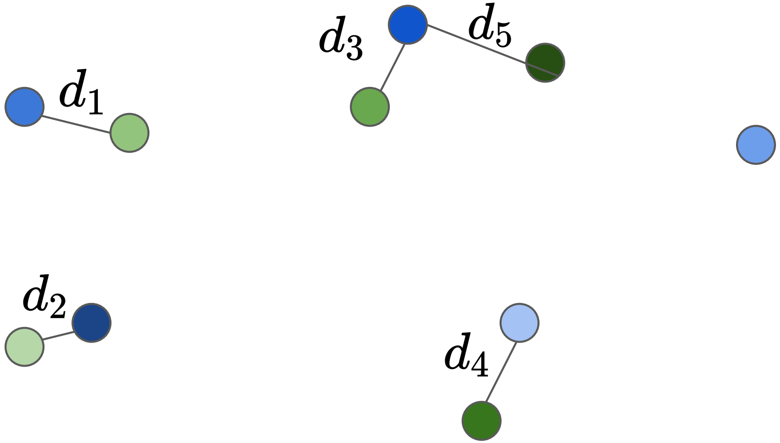
Thibault Groueix, Pierre-Alain Langlois, 2019

Loss on pointclouds

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



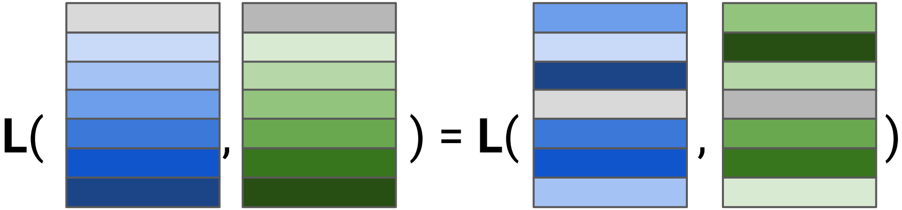
$$= \frac{1}{5} \cdot (d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2)$$



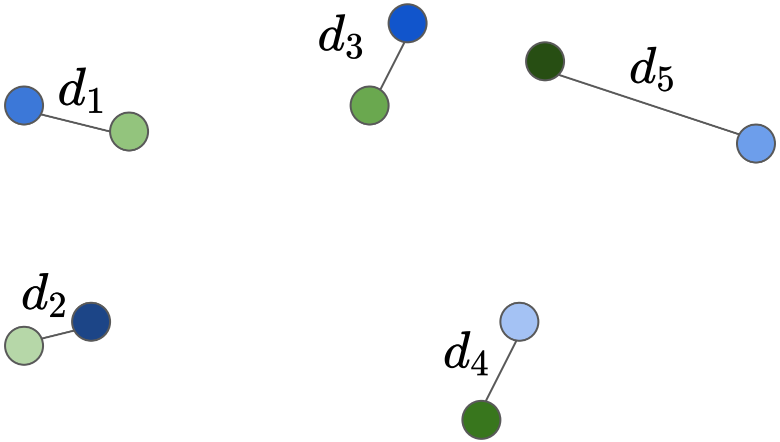
	Complexity
EMD	n^3
Chamfer	n^2

Loss on pointclouds

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



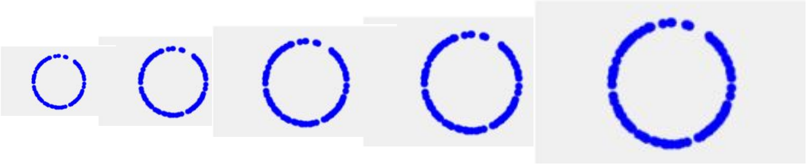
$$= \frac{1}{5} \cdot (d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2)$$



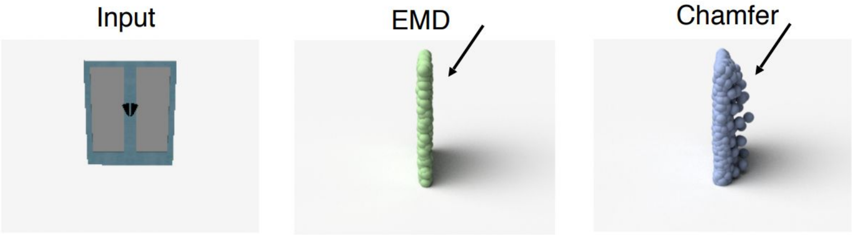
	Complexity
EMD	n^3
Chamfer	n^2

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

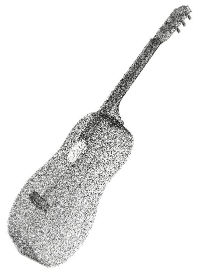
Choice of representation

PointClouds (PointSetGen)

Loss : Chamfer Distance

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

3D Object **Y**



Thibault Groueix, Pierre-Alain Langlois, 2019

Generating points

Encoder

E

Decoder

D

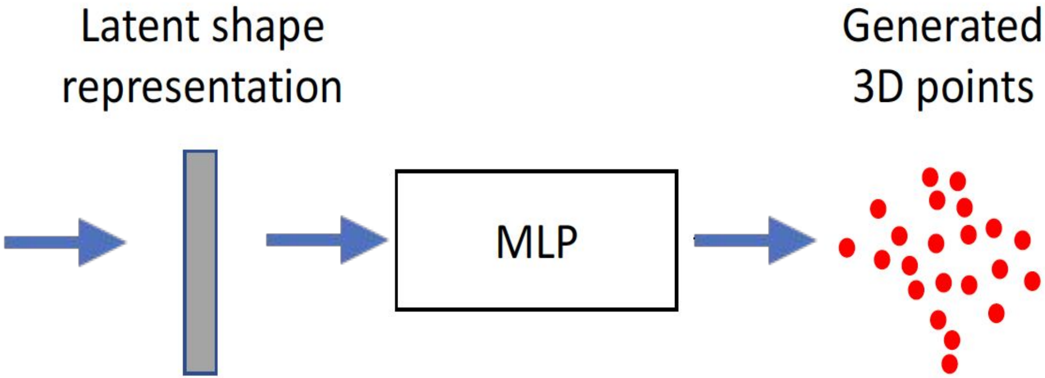
Test Shape

Generating points

Encoder
E

Decoder
D

Test Shape

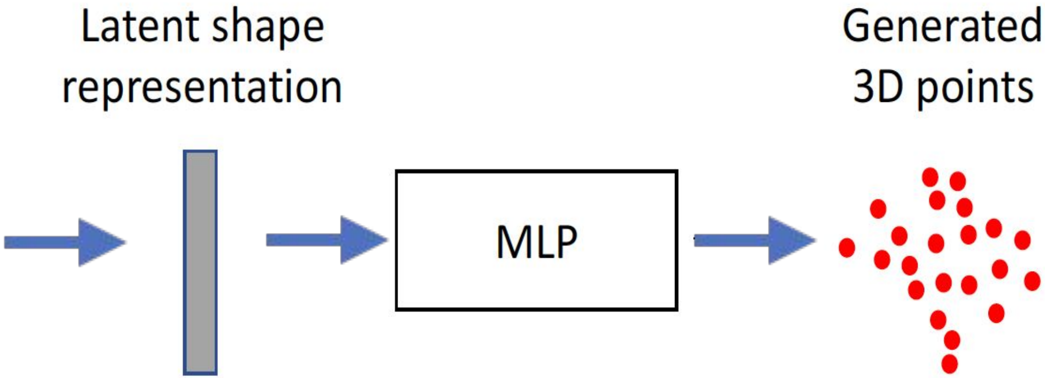


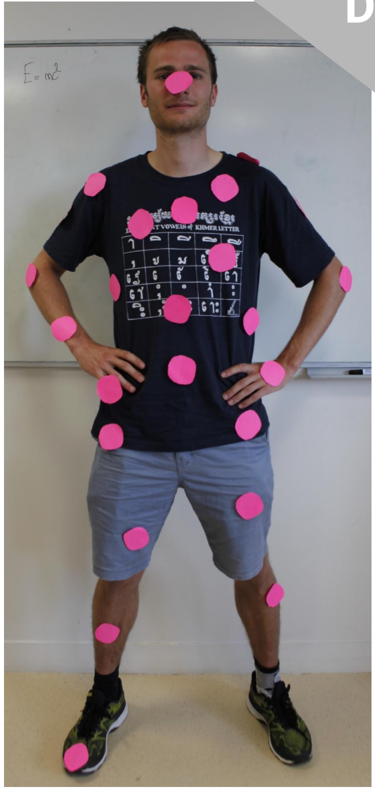
Generating points

Encoder
E

Decoder
D

Test Shape

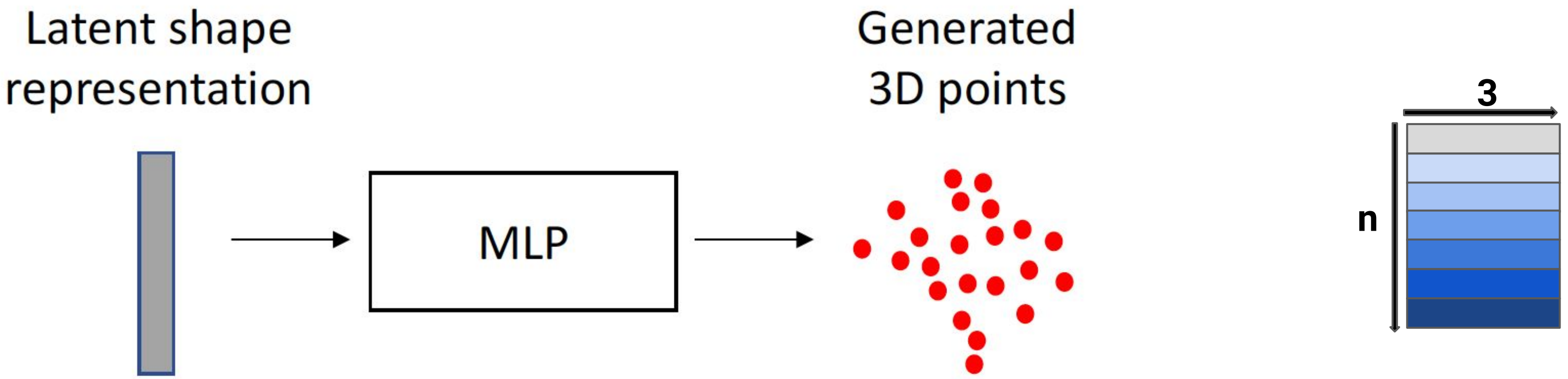




Thibault Groueix, Pierre-Alain Langlois, 2019

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

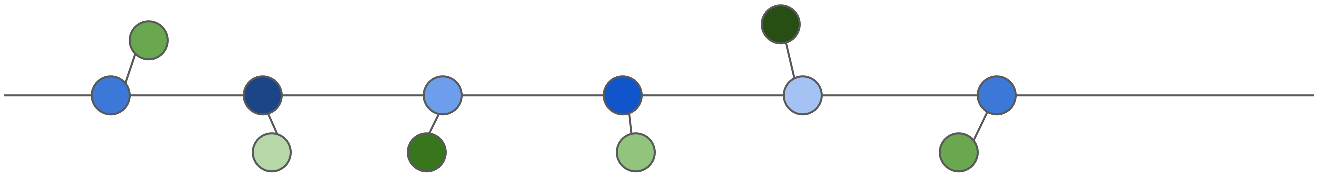


Limitation of PointSetGen [Fan2017]

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

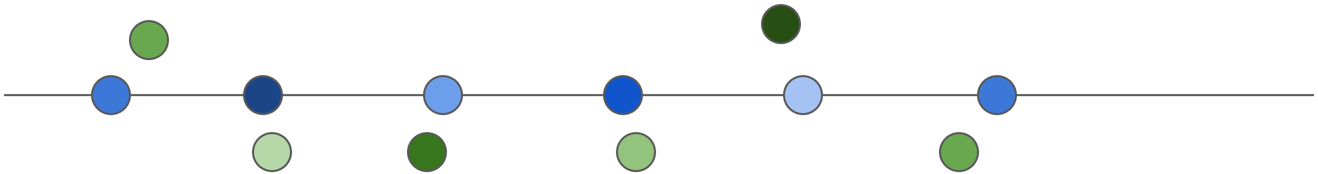
Limitation of PointSetGen [Fan2017]

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



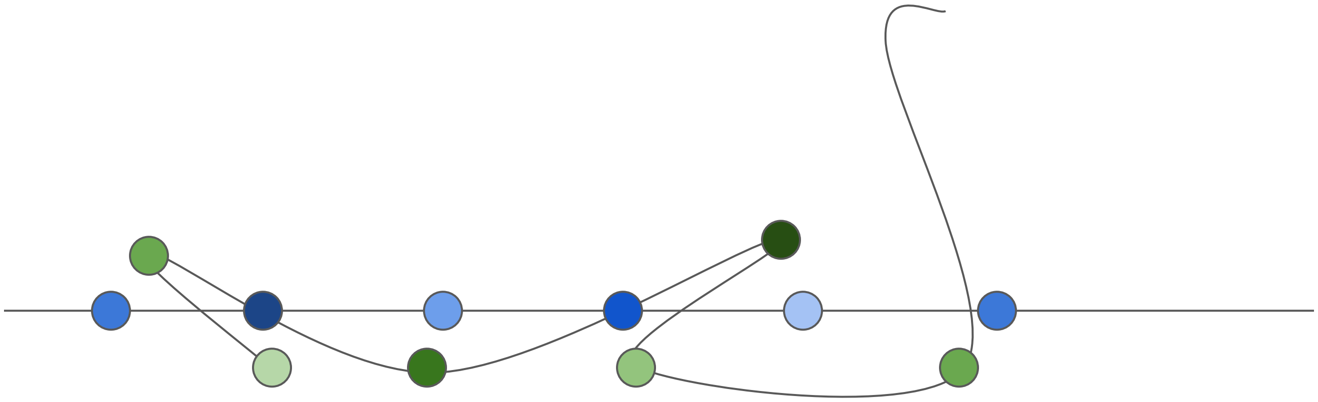
Limitation of PointSetGen [Fan2017]

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



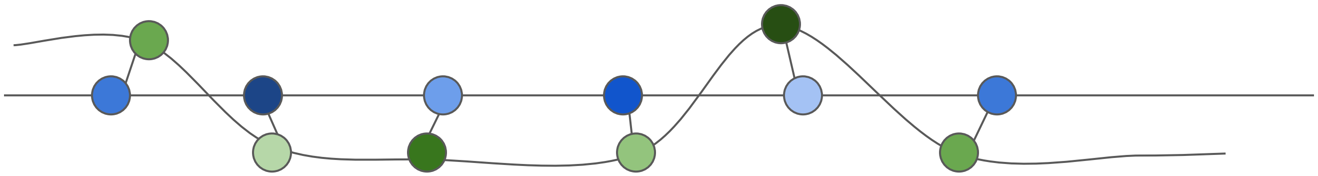
Limitation of PointSetGen [Fan2017]

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



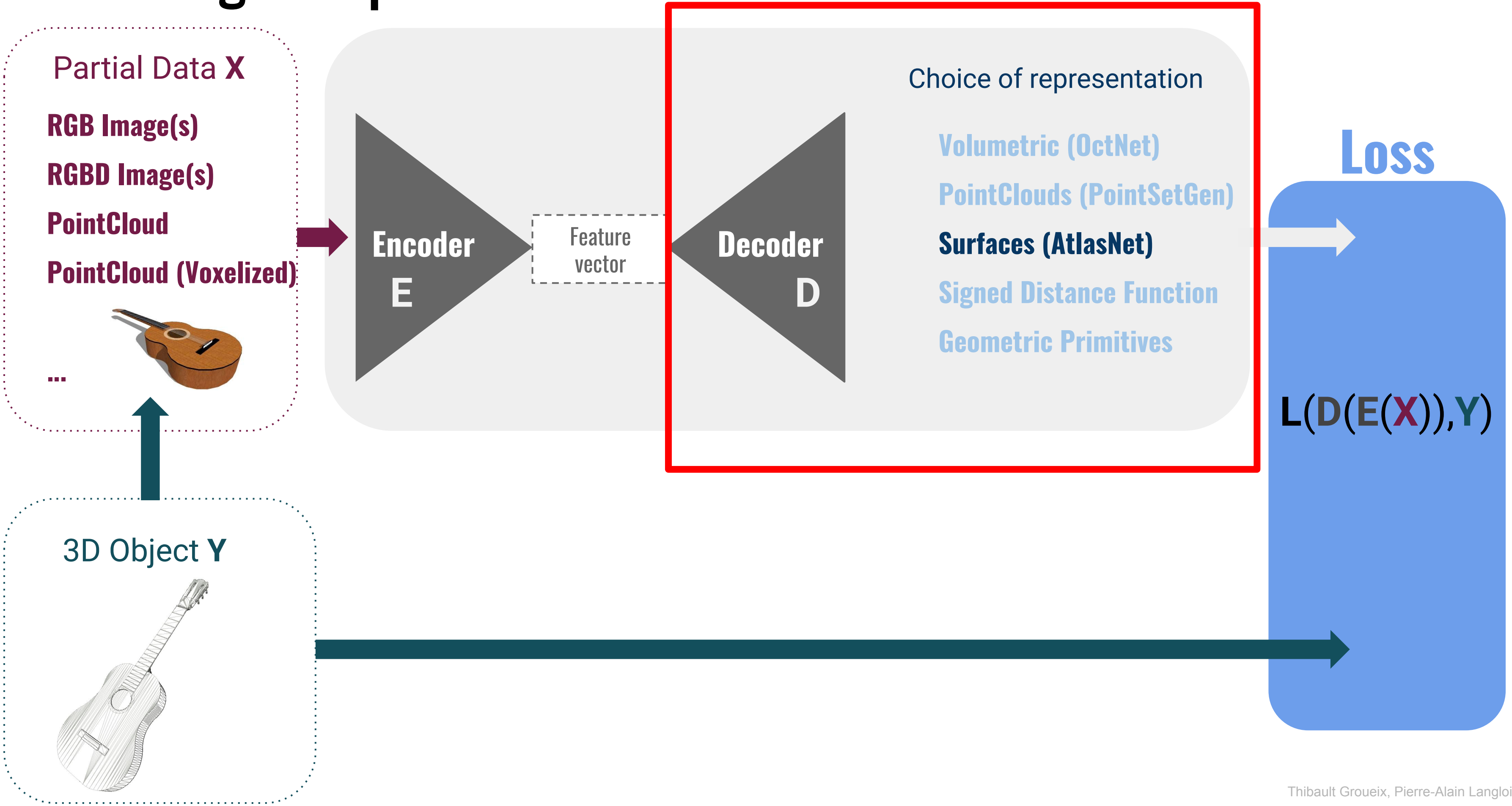
Limitation of PointSetGen [Fan2017]

- ➔ 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**]

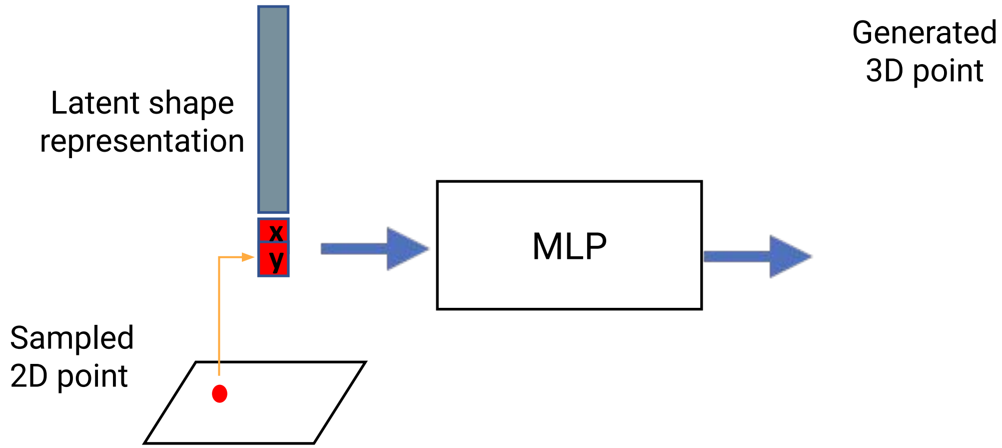
Training setup for 3D reconstruction



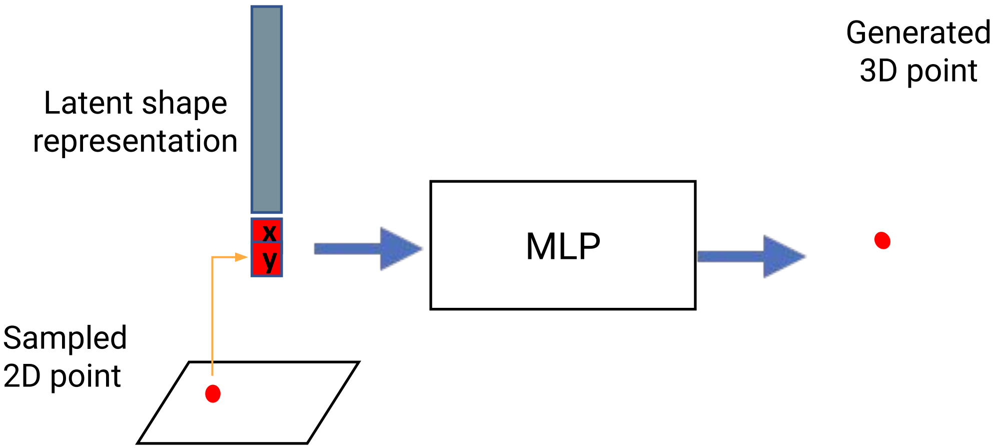
Thibault Groueix, Pierre-Alain Langlois, 2019

Deform a surface [Groueix2018]

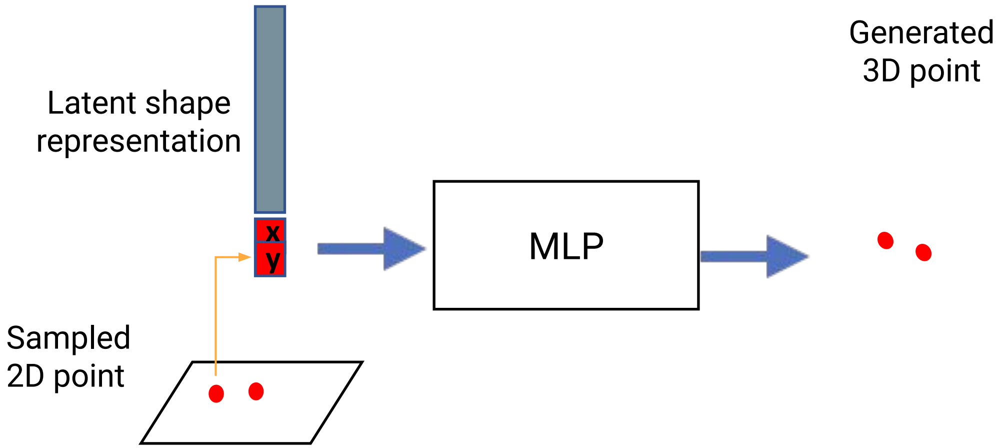
Decoder D



Deform a surface : space mapping trick [Groueix2018]

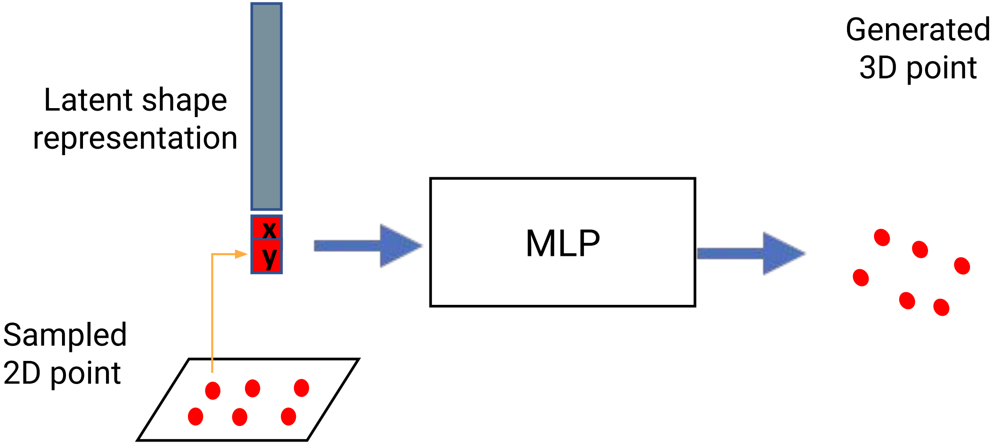


Deform a surface [Groueix2018]

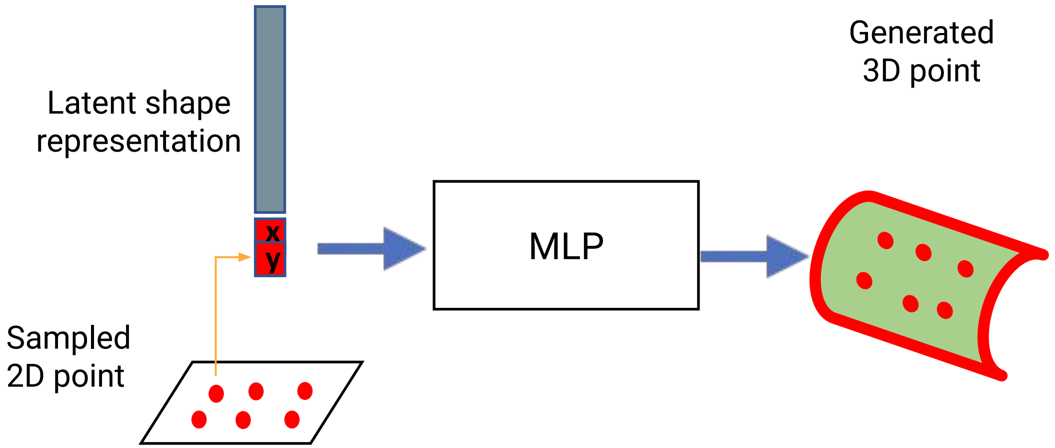


Deform a surface [Groueix2018]

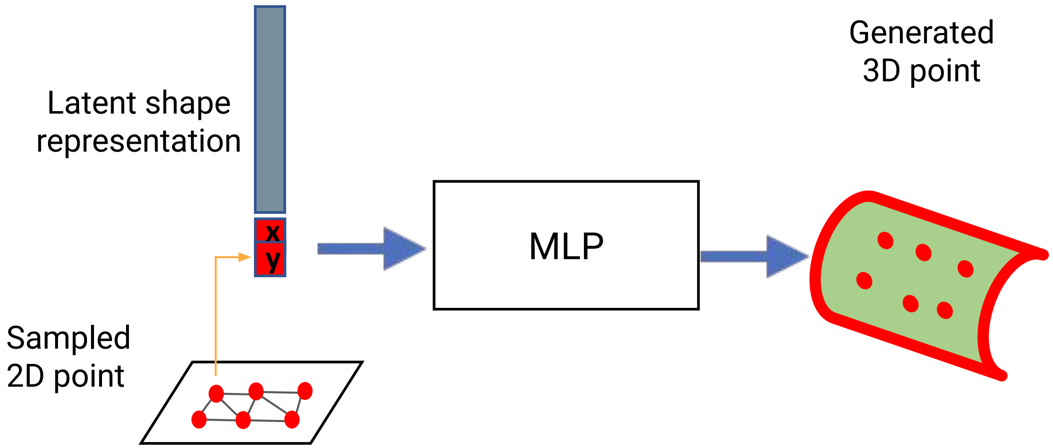
Decoder D



Deform a surface [Groueix2018]

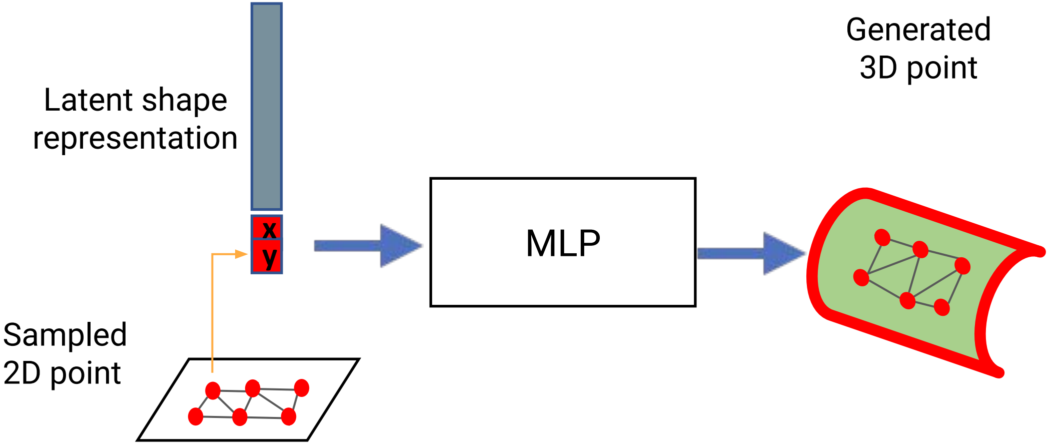


Deform a surface [Groueix2018]



Deform a surface [Groueix2018]

Decoder
D

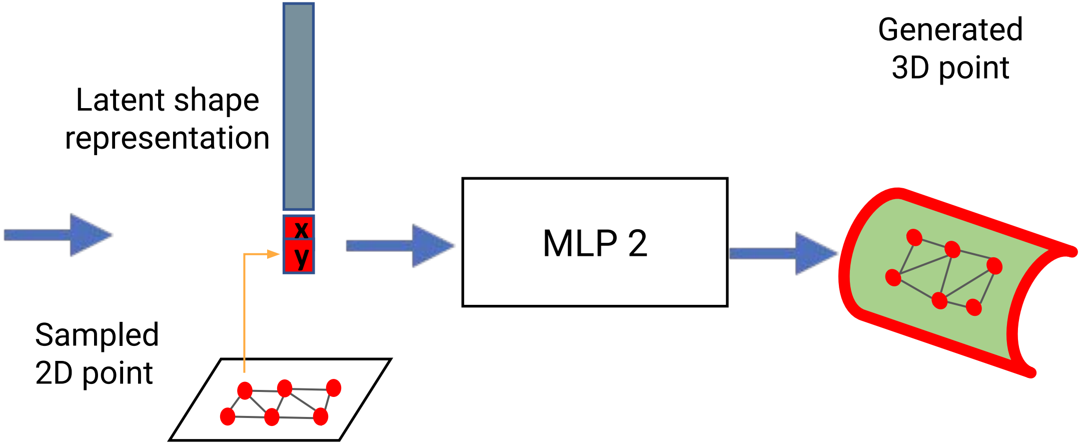


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape

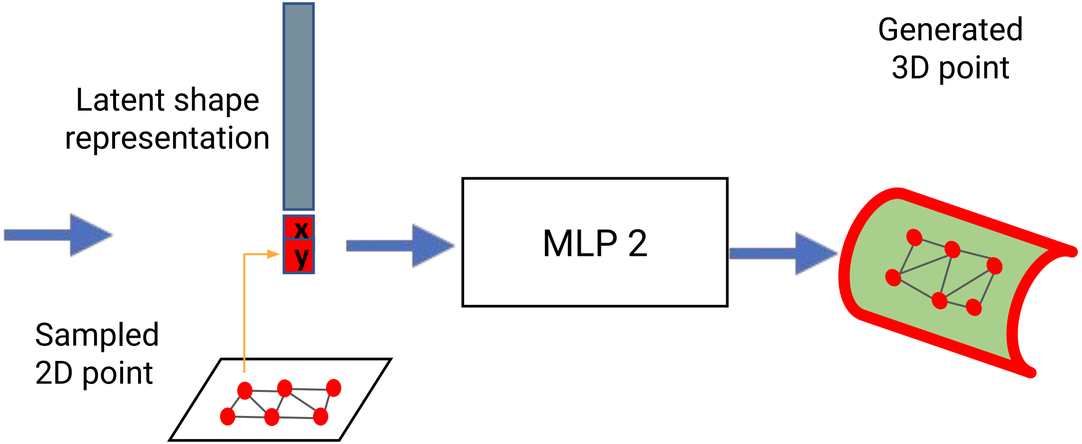


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape

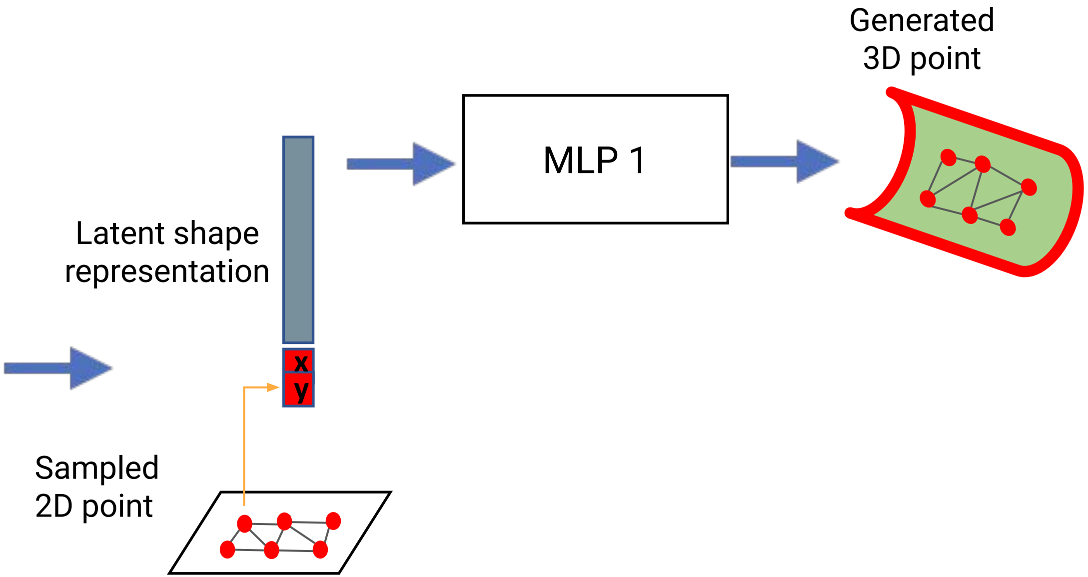


Deform a surface [Groueix2018]

Encoder
E

Decoder
D

Test Shape

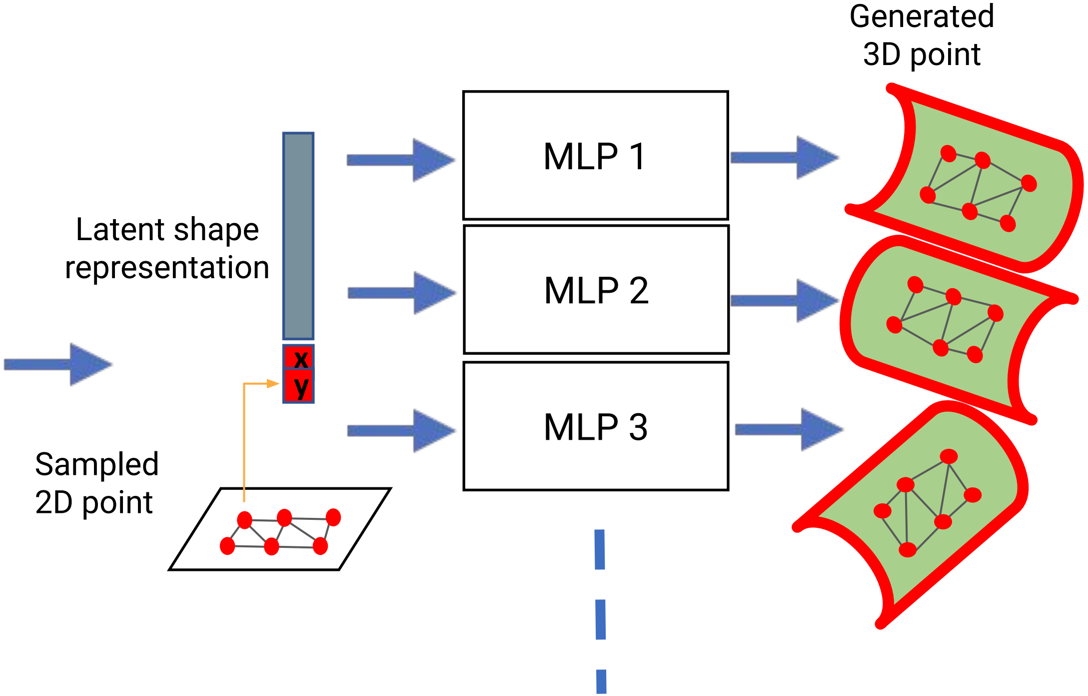


Deform a surface [Groueix2018]

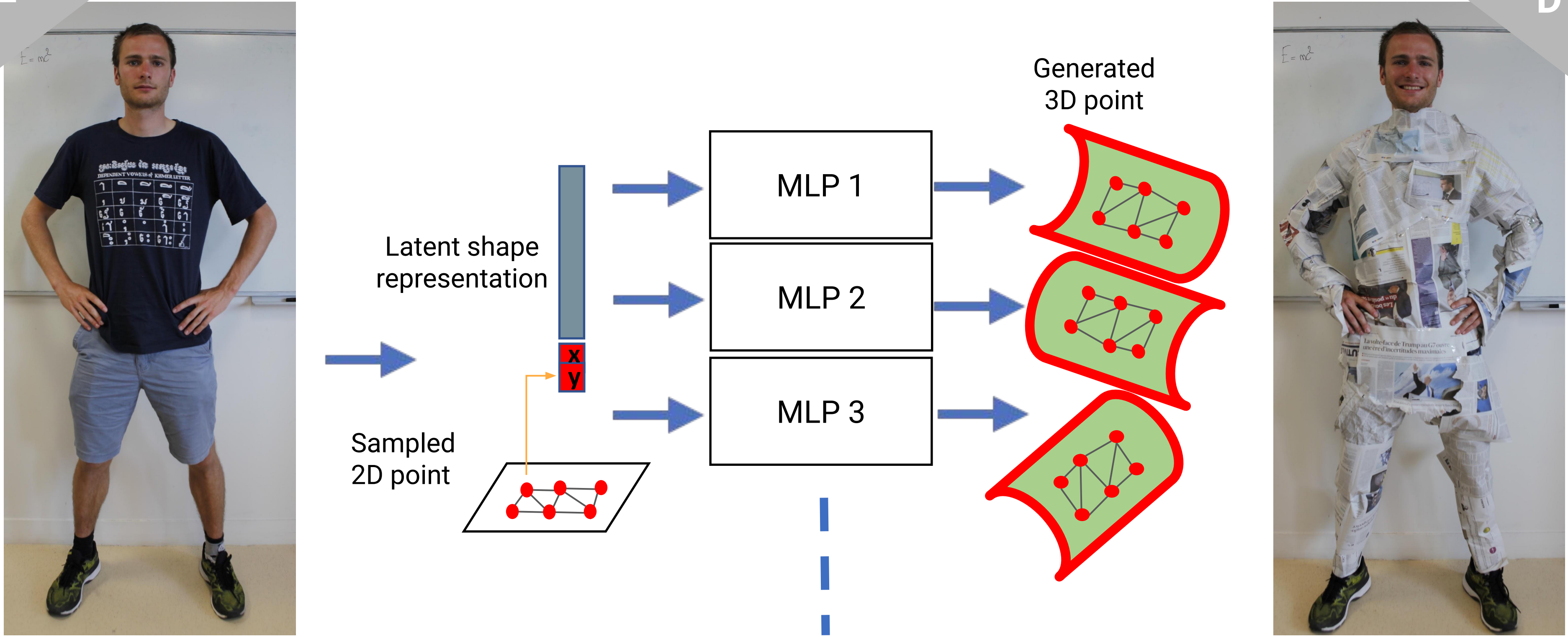
Encoder
E

Decoder
D

Test Shape

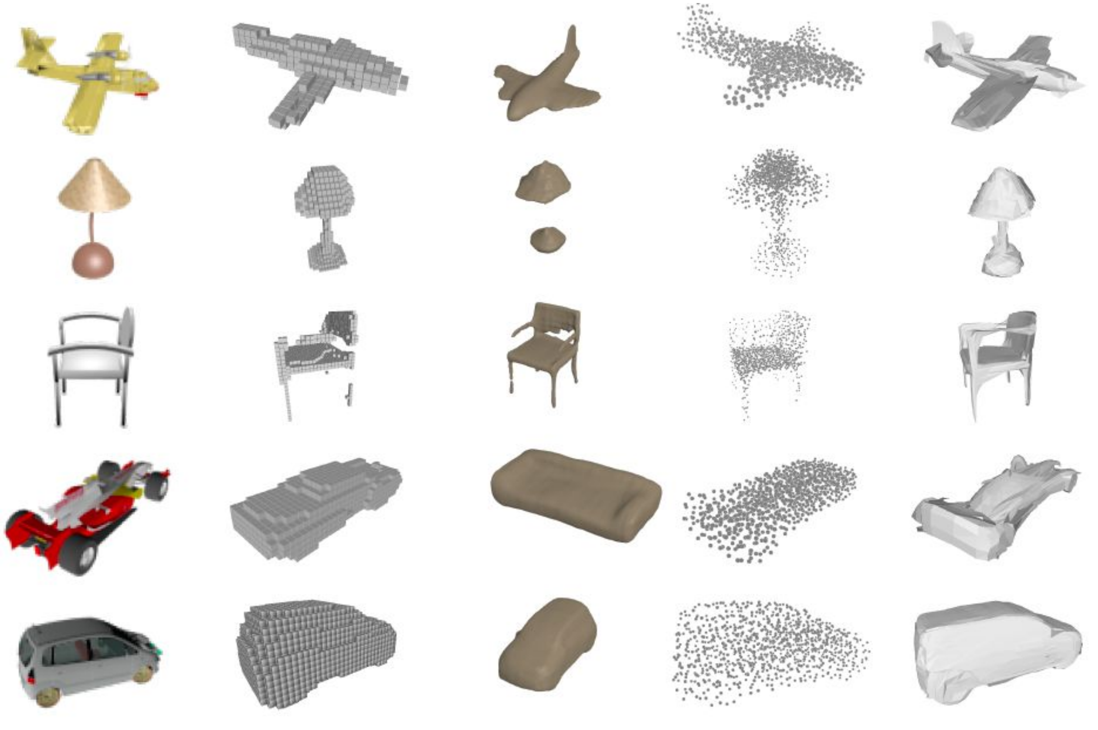


Deform a surface [Groueix2018]



Test Shape

Results : Single View Reconstruction

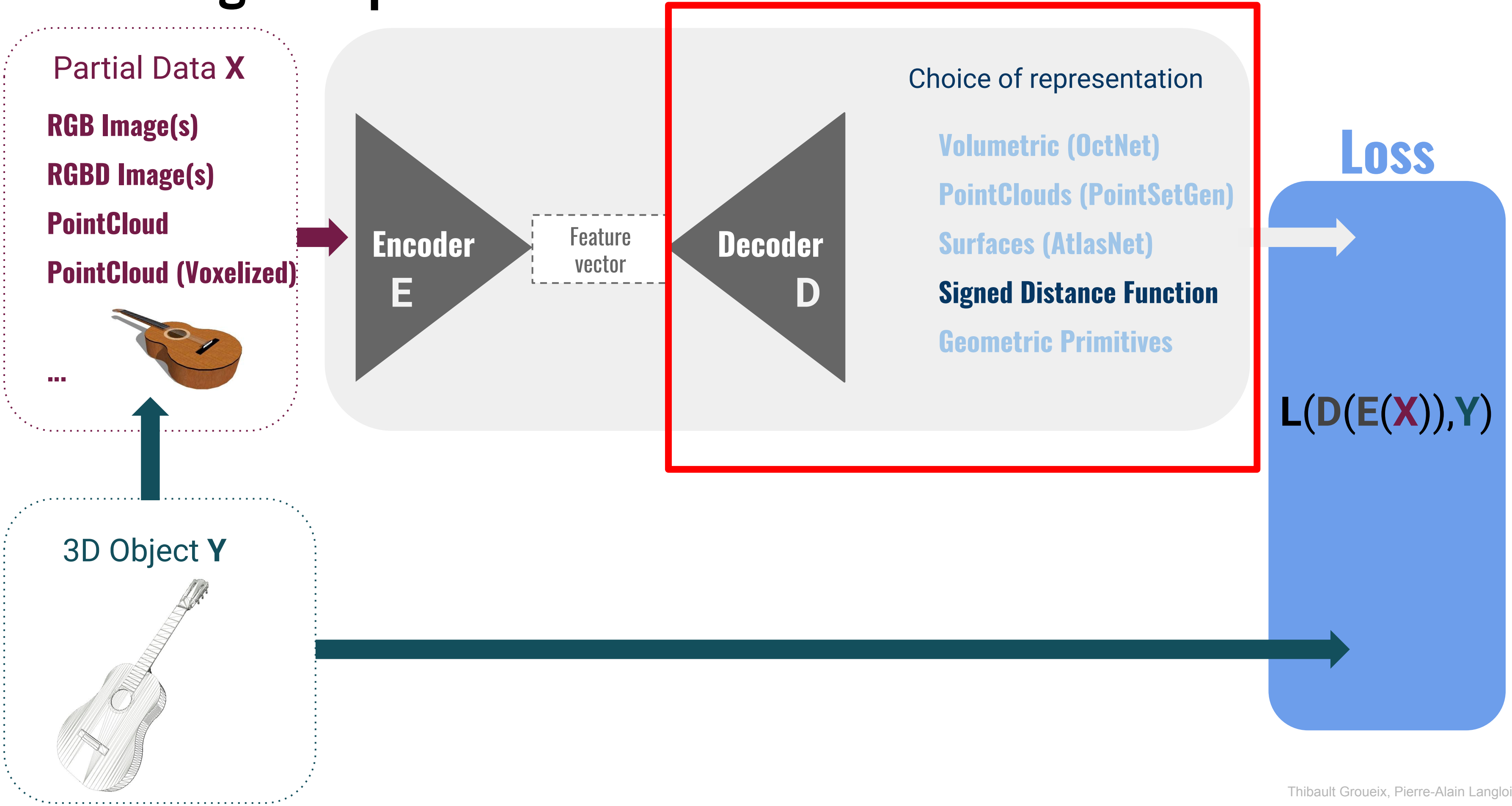


(a) Input (b) 3D-R2N2 (c) HSP (d) PSG (e) Ours

Direct application : mesh parametrization

State-of-the-art correspondences of FAUST [Groueix2018b]

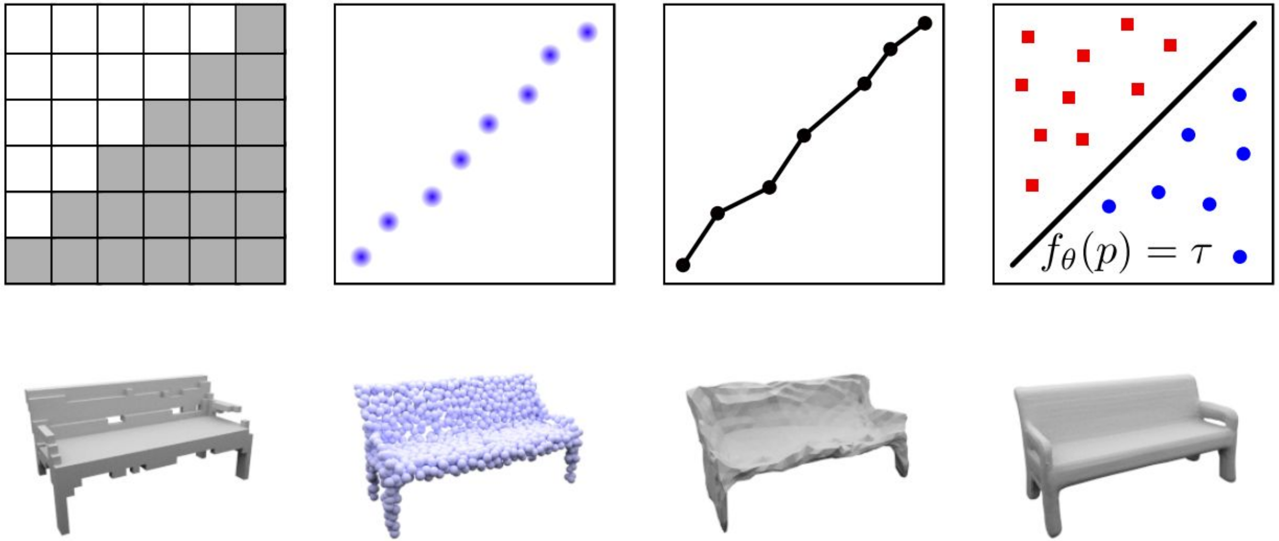
Training setup for 3D reconstruction

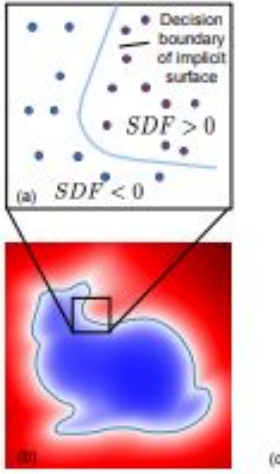


Thibault Groueix, Pierre-Alain Langlois, 2019

Can the space mapping trick be applied to volumetric representations ?

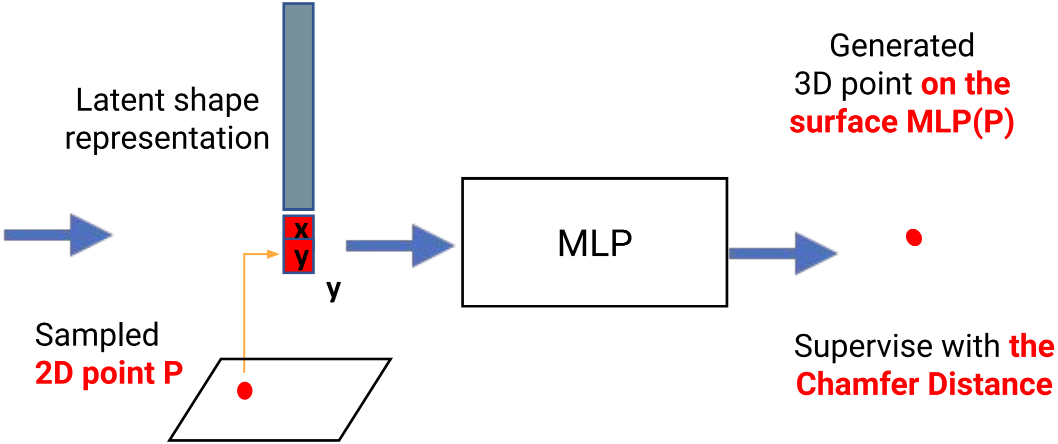
-> yes, through the Signed Distance Function (SDF) ! [Mescheder2018], [Park2019], [Chen2019]





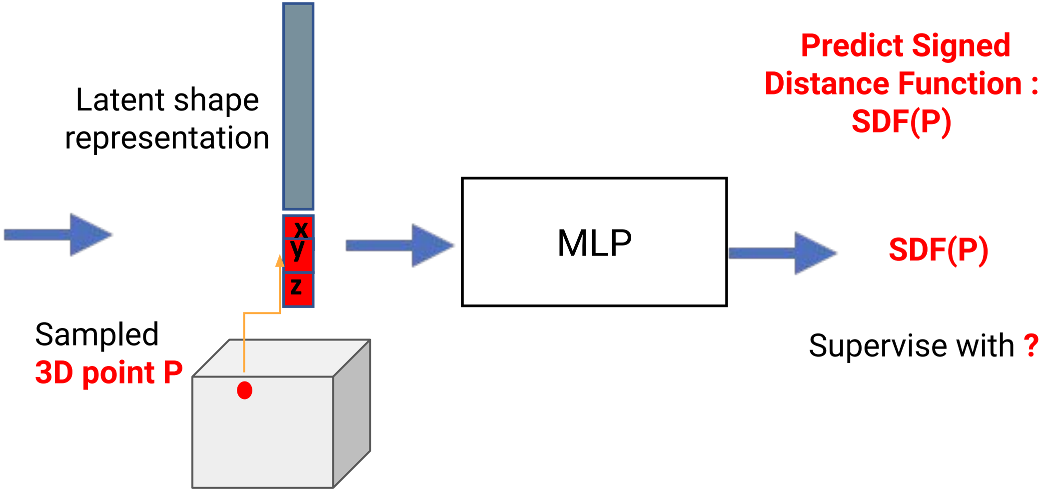
Credit:[Mescheder2018] a) Voxels b) Points c) Meshes d) Signed Distance Function

Deform a surface : space mapping trick [Groueix2018]



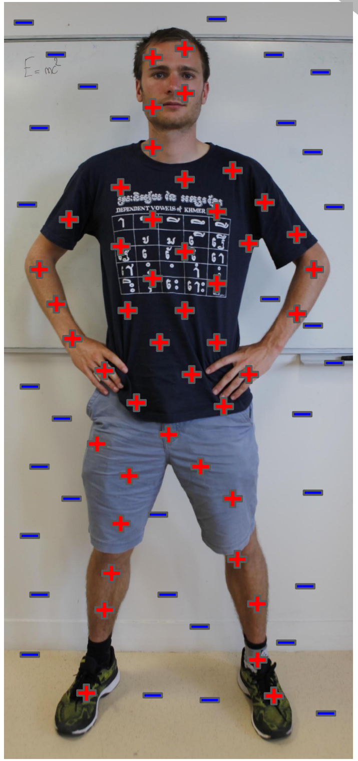
Deform a volume [Mescheder2018]

Decoder D

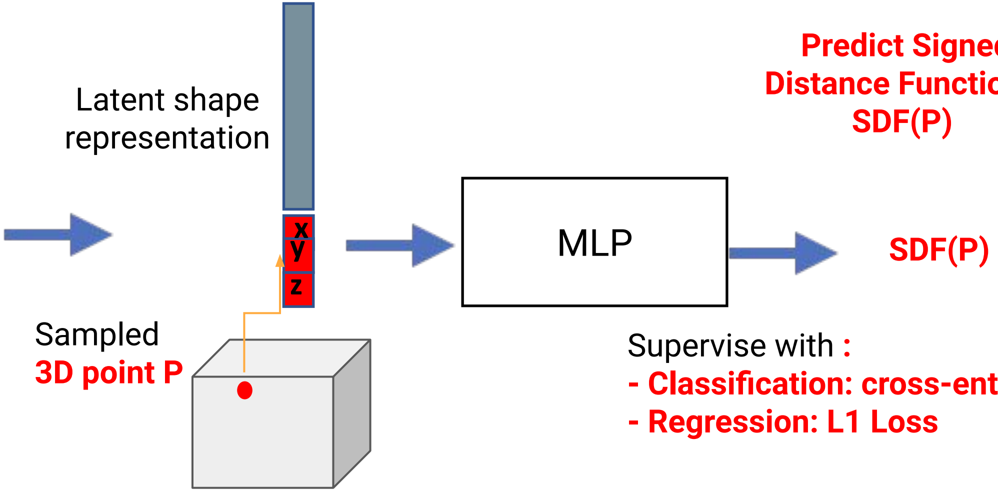


Predict Signed Distance Function : SDF(P)

Supervise with ?

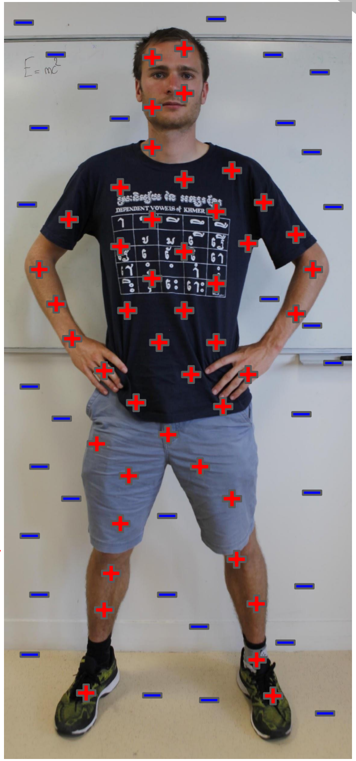


Deform a volume [Mescheder2018]



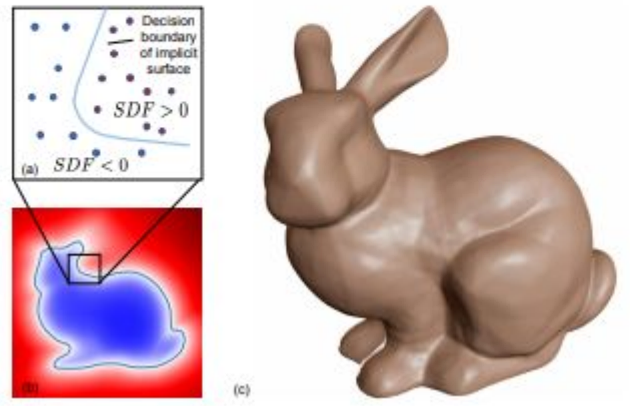
Predict Signed Distance Function : SDF(P)

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]

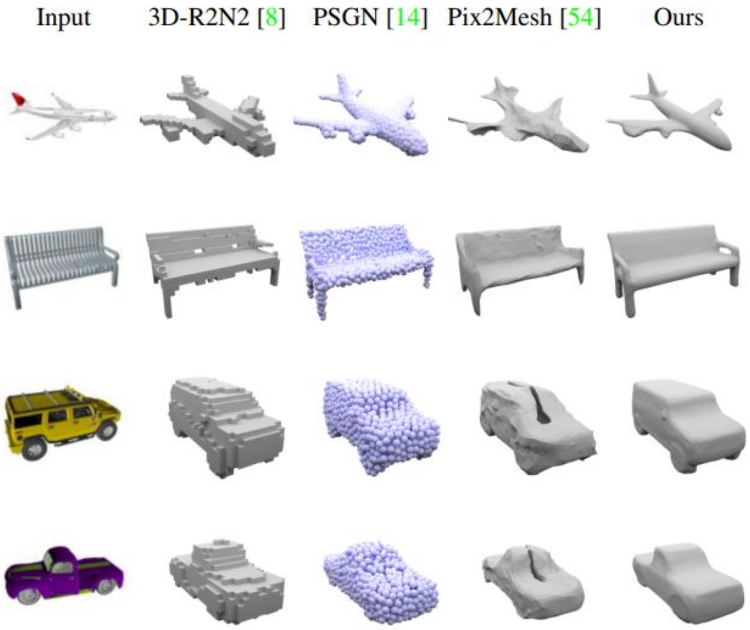
++ Get a voxel based representation at infinite granularity

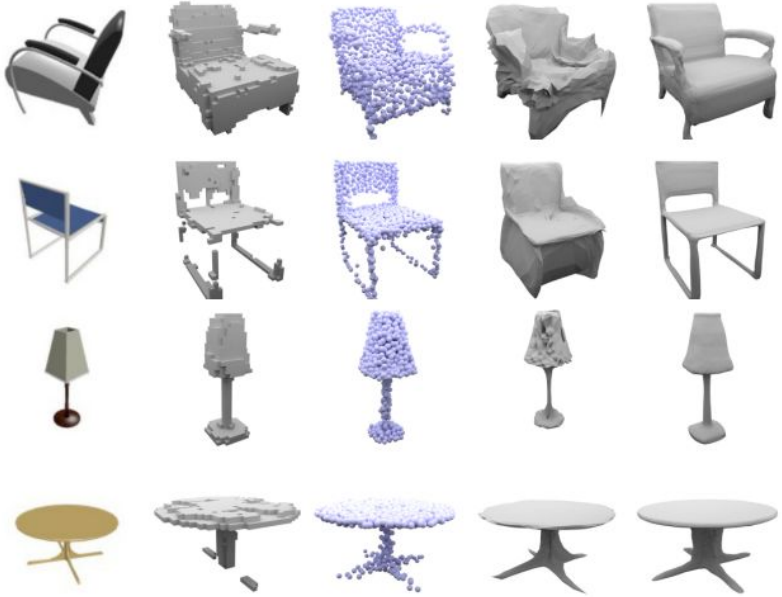
++ Get analytic normals : dSDF(x)/dx

++ Topology is no longer an issue

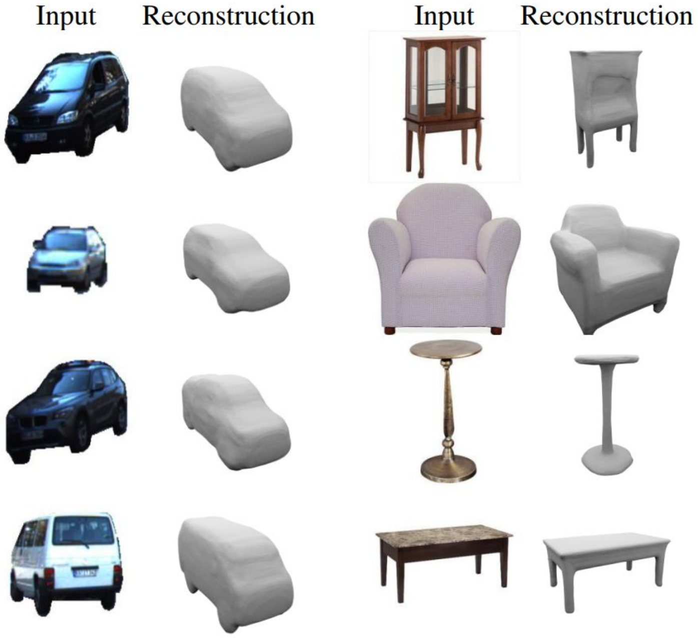
-- need only one assumption : there is an interior and an exterior

Single View Reconstruction Results [Mescheder2018]



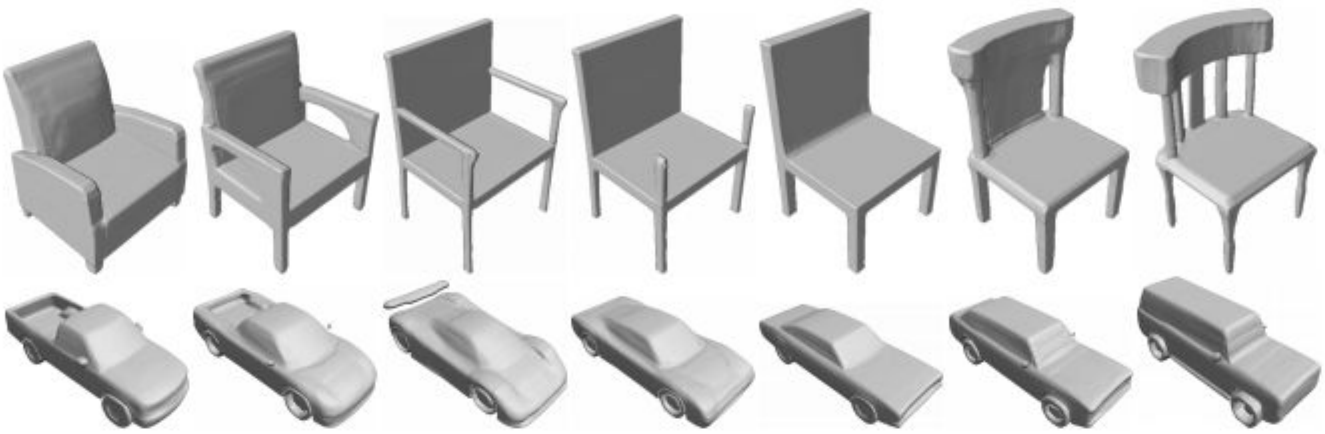


Test on Real Images [Mescheder2018]



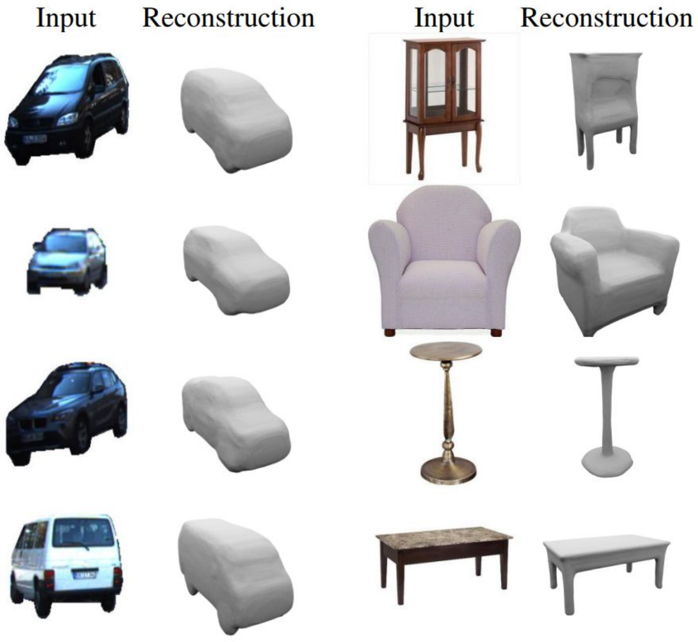
(a) KITTI (b) Online Products

Interpolation Results [Park2019]



Limitations so far - SDF

Reconstructed models are too smooth



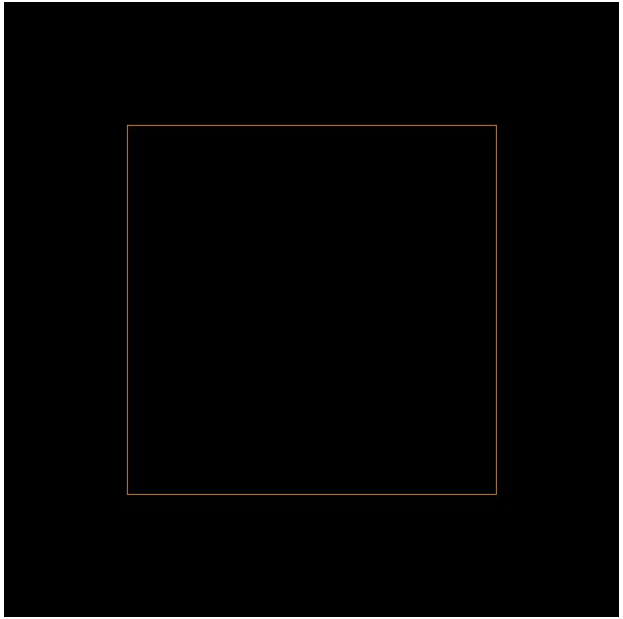
(a) KITTI (b) Online Products

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

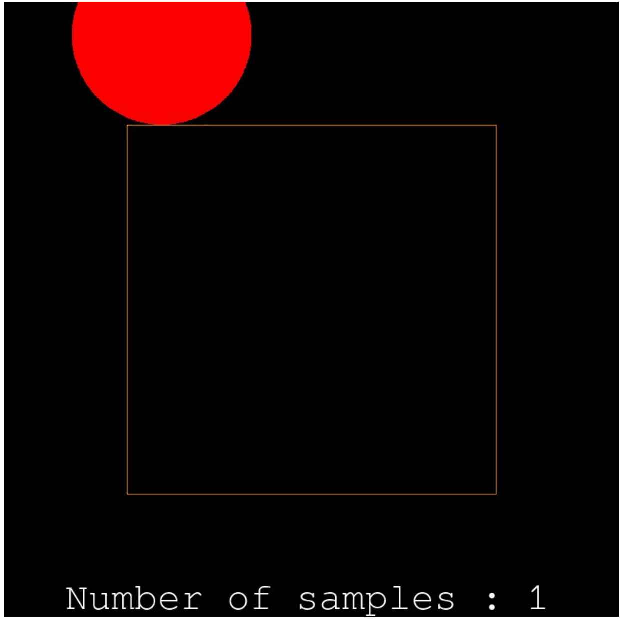


Limitations so far - SDF

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

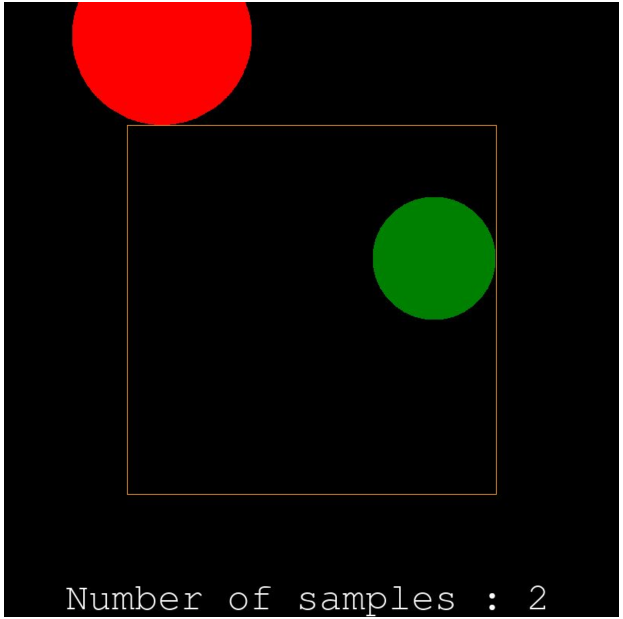


Limitations so far - SDF

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

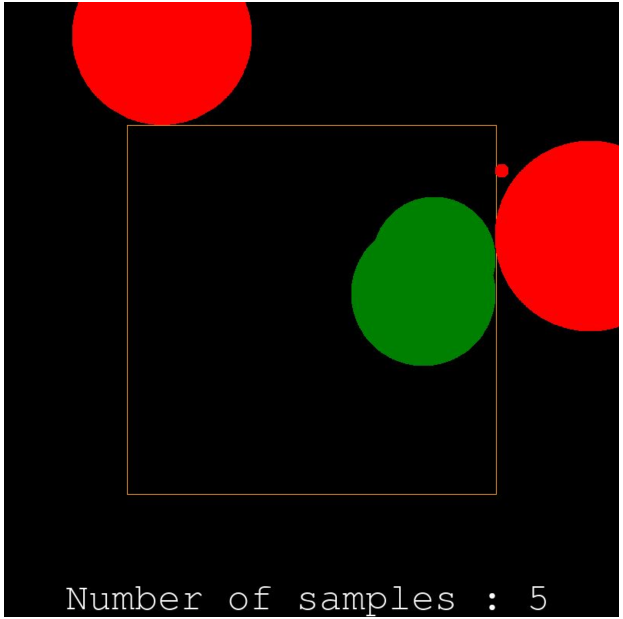


Limitations so far - SDF

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

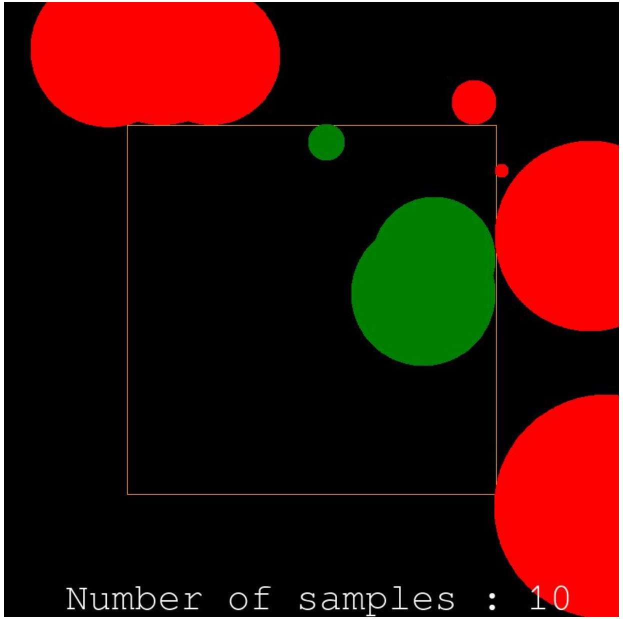


Limitations so far - SDF

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

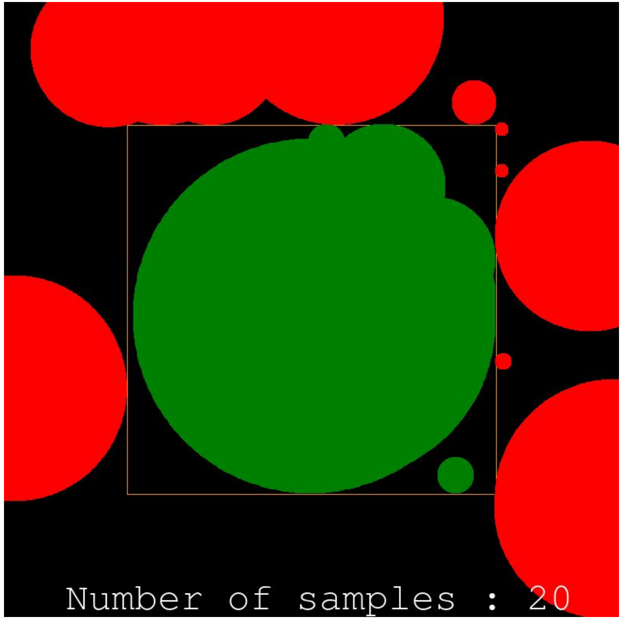
Decoder

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

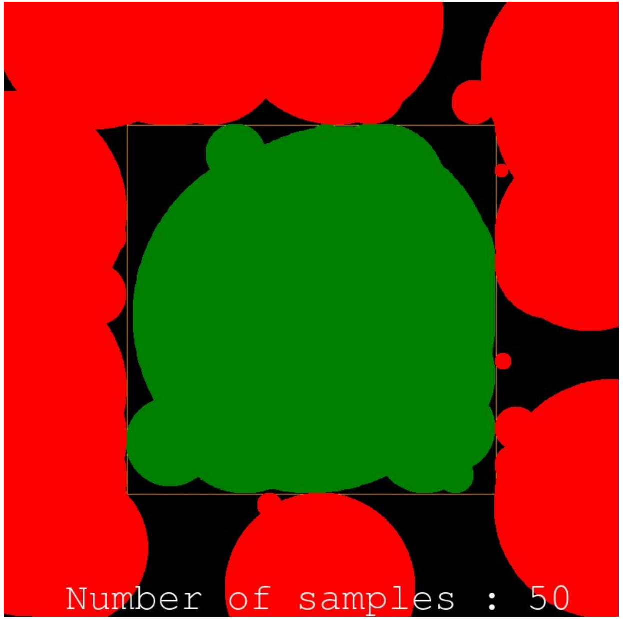


Limitations so far - SDF

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

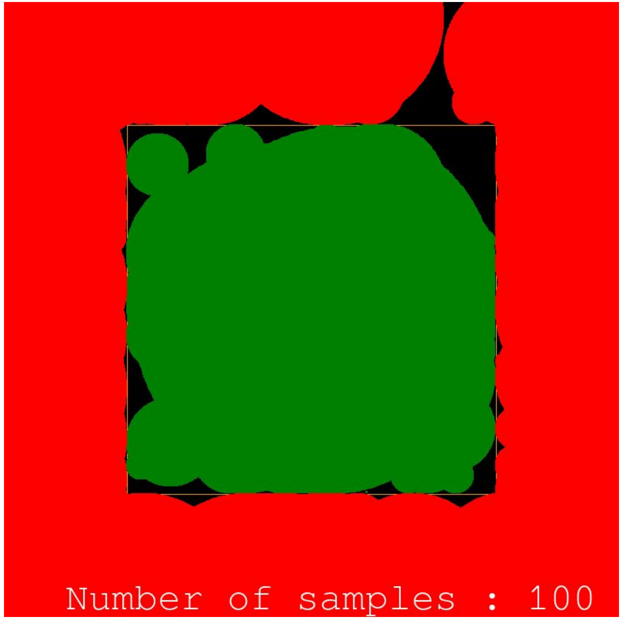


Limitations so far - SDF

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

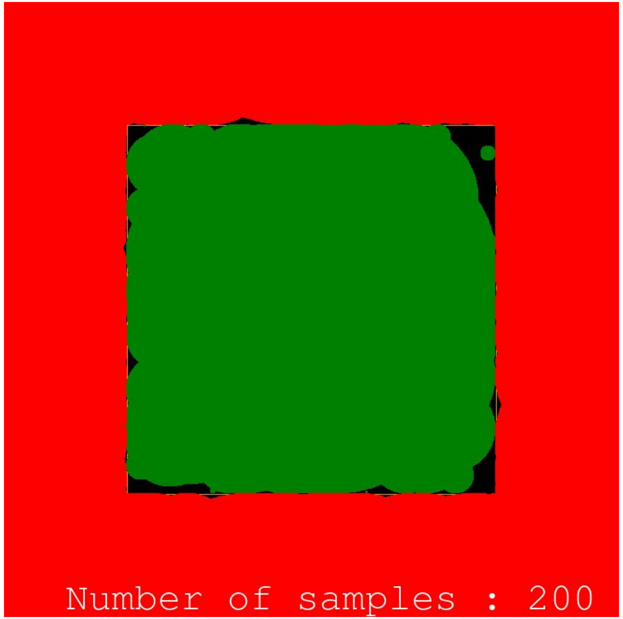


Limitations so far - SDF

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

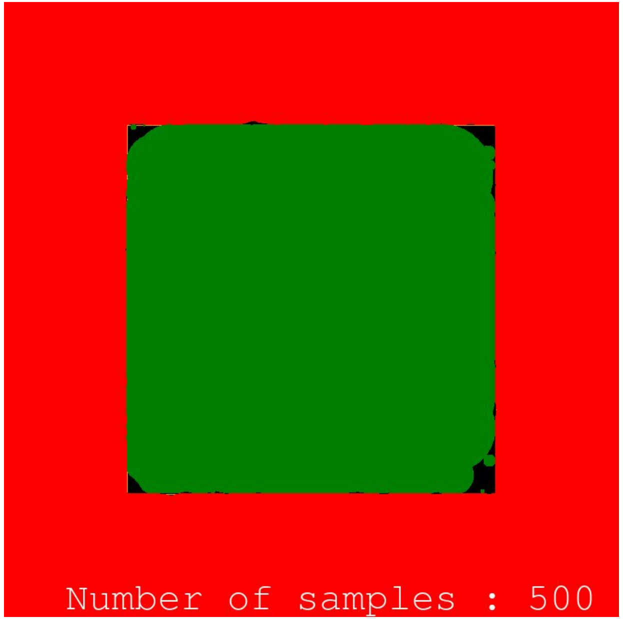


Limitations so far - SDF

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

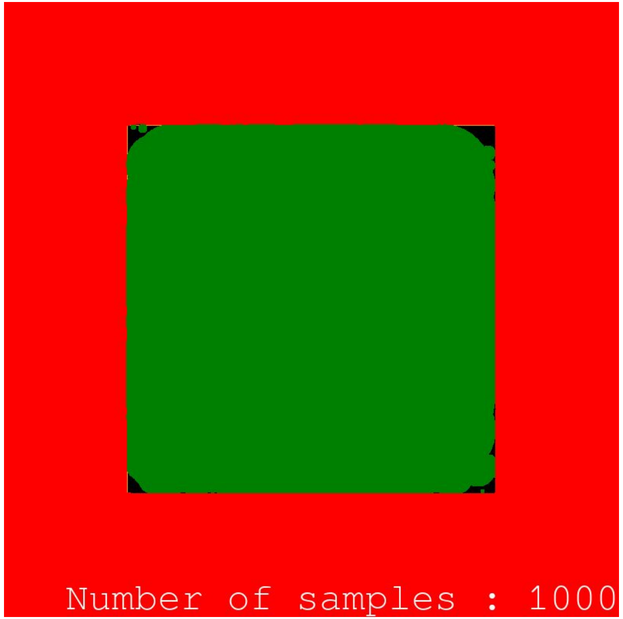


Limitations so far - SDF

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



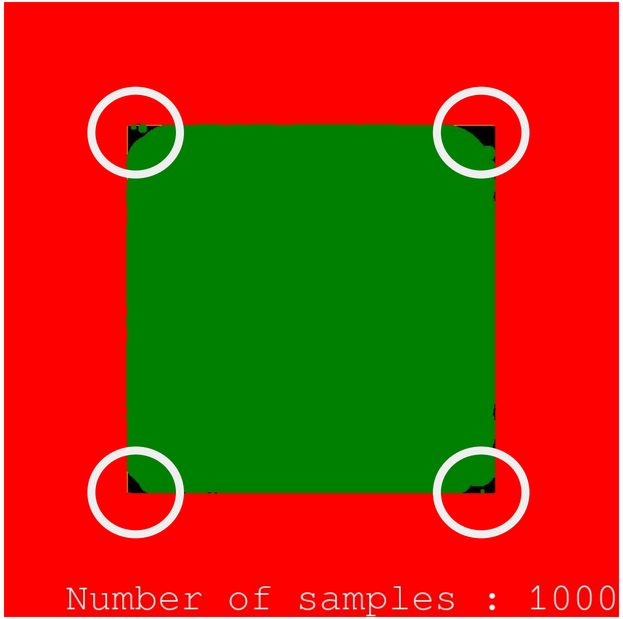
Limitations so far - SDF

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

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



Limitations so far - SDF

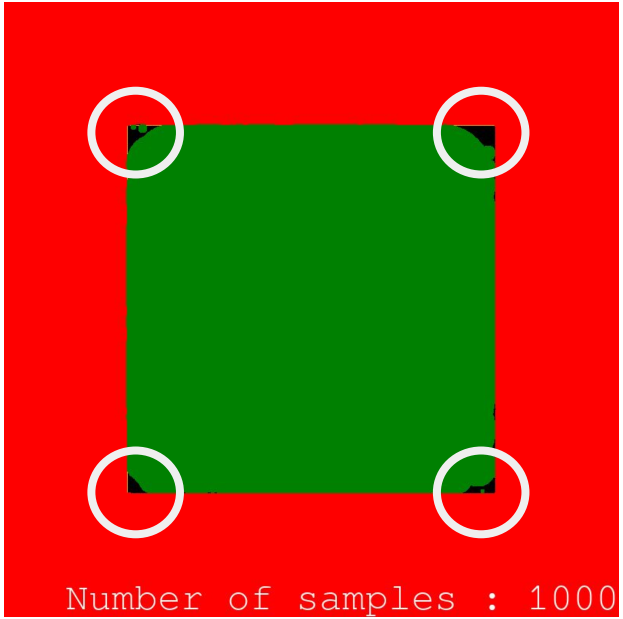
Reconstructed models are too smooth - Possible explanation : Monte-Carlo sampling

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

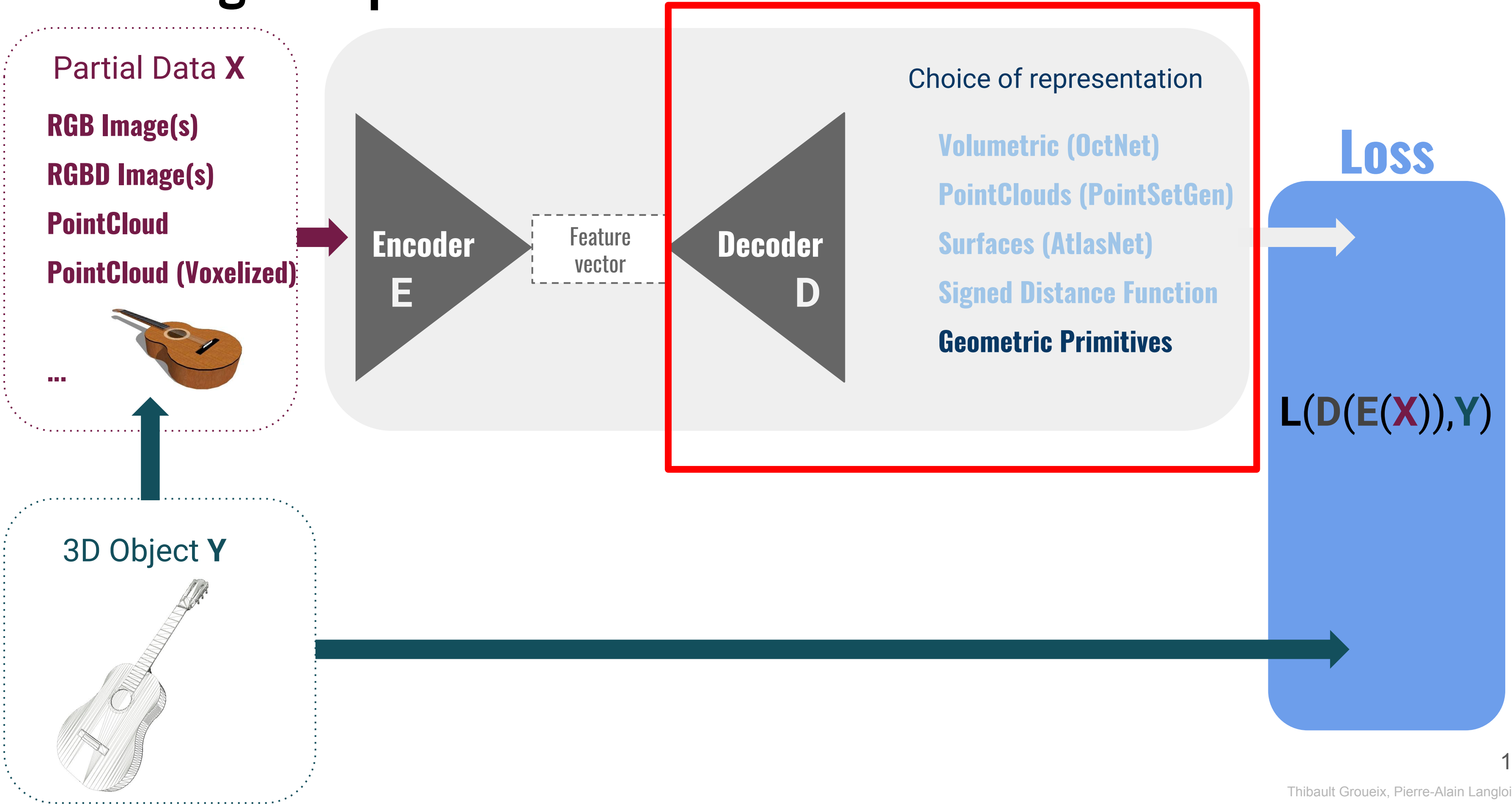
- If r > 0, the circle (p, r) is full
- If r < 0, the circle (p, -r) is empty

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

Potential fix : non uniform sampling



Training setup for 3D reconstruction



Fitting geometric primitives to a 3D shape

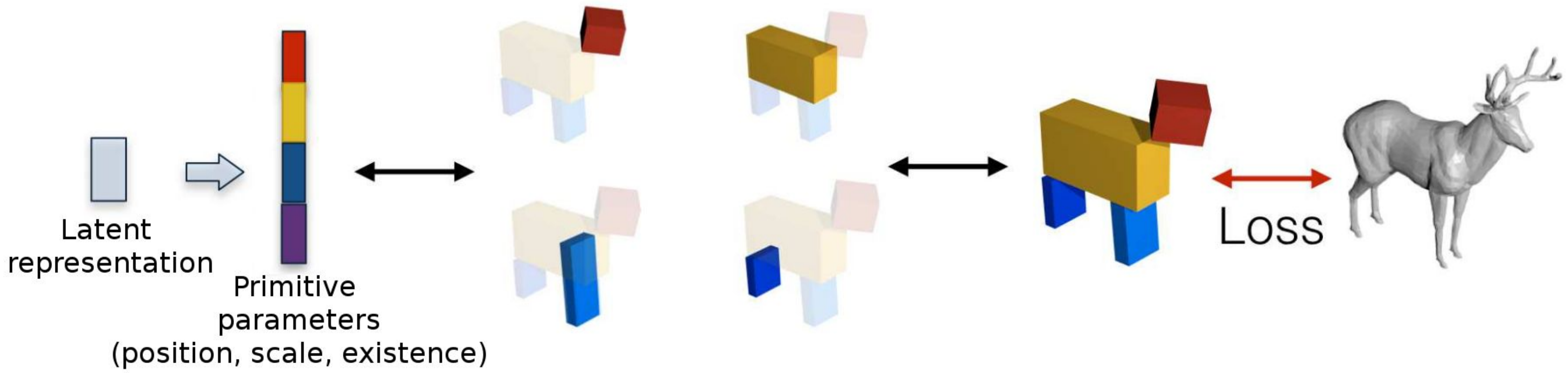
Everything in nature takes its form from the sphere, the cone and the cylinder.
- Paul Cezanne.

Motivations :

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

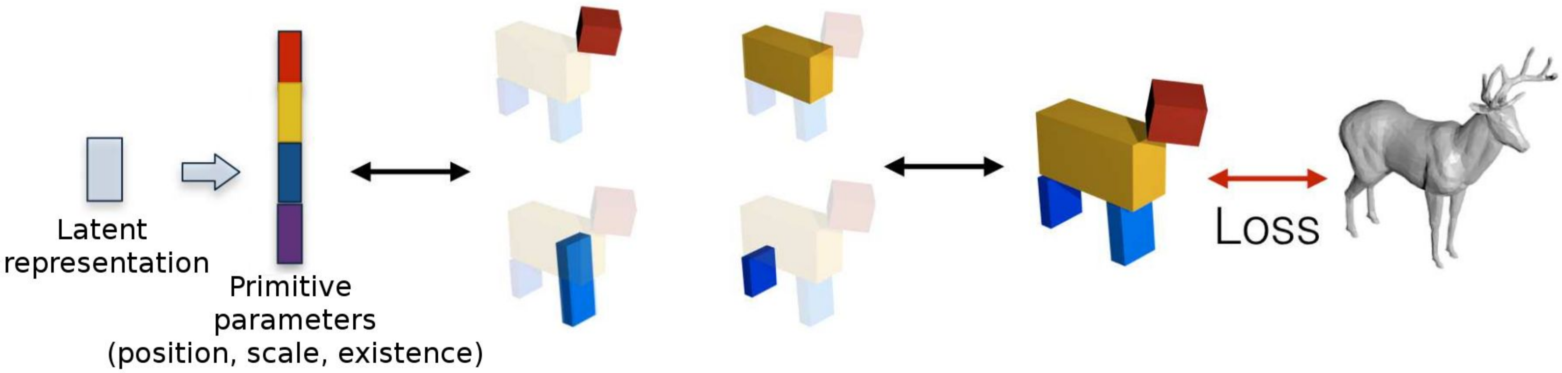
Learn 3D reconstruction with cuboids [Tulsiani2017]

Unsupervised method for fitting cuboid primitives



Learn 3D reconstruction with cuboids [Tulsiani2017]

Unsupervised method for fitting cuboid primitives

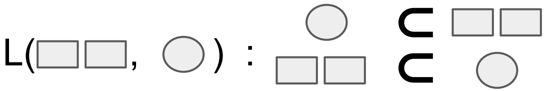


Challenges :

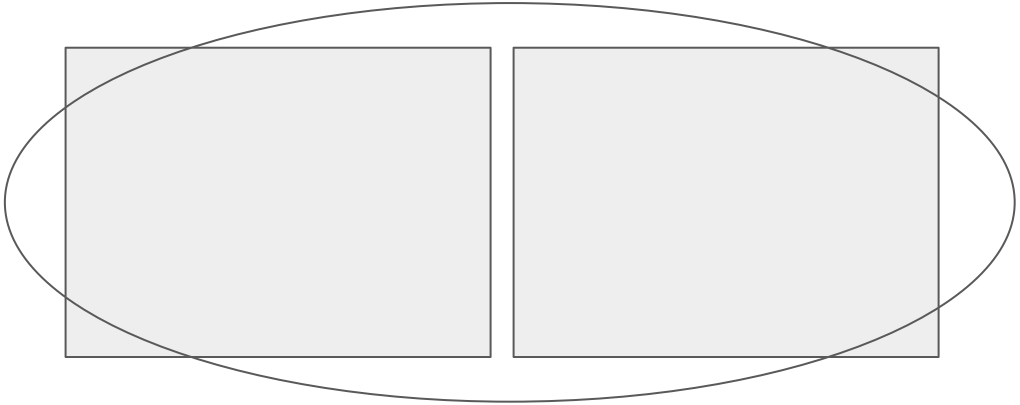
1. Position the cuboids
2. Estimating the amount of cuboids to predict

Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss : Chamfer ?

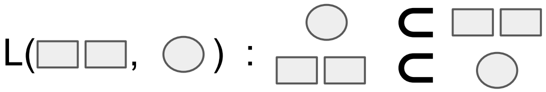


Problem !

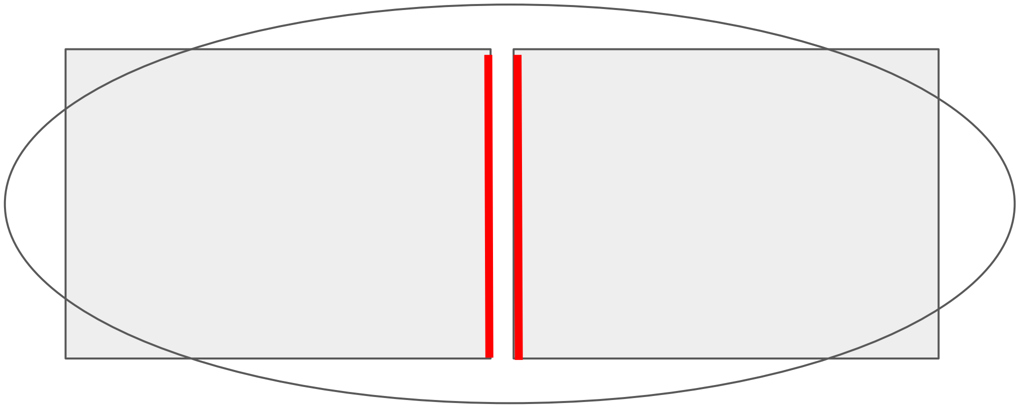


Fitting cuboids to a 3D shape [Tulsiani2017]

Designing a loss : Chamfer ?



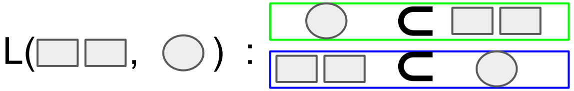
Problem !



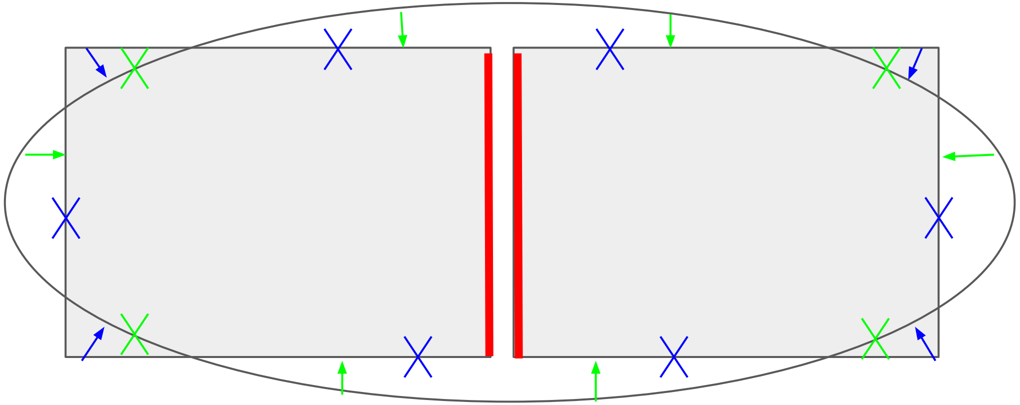
➔ Points sampled on increase the Chamfer distance

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]

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 θ_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 Θ_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 Θ_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 Θ_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 Θ_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 Θ_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 Θ_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]

Decoder
D

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

$$\begin{aligned}
\frac{\partial}{\partial \theta}\mathbb{E}[f(X)] &= \frac{\partial}{\partial \theta} \sum_{x \in \mathcal{X}} f(x) p_\theta(x) \\
&= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} p_\theta(x) \\
&= \sum_{x \in \mathcal{X}} f(x) \frac{\partial}{\partial \theta} \log(p_\theta(x)) p_\theta(x) \\
&= \mathbb{E}\left[f(X) \frac{\partial}{\partial \theta} \log(p_\theta(X)) \right]
\end{aligned}$$

"log-likelihood trick"

Estimating the amount of cuboids to predict [Tulsiani2017]

Decoder
D

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

$$\begin{aligned}
\frac{\partial}{\partial \theta}\mathbb{E}[f(X)] &= \frac{\partial}{\partial \theta}\sum_{x\in\mathcal{X}} f(x)p_\theta(x) \\
&= \sum_{x\in\mathcal{X}} f(x)\frac{\partial}{\partial \theta}p_\theta(x) \\
&= \sum_{x\in\mathcal{X}} f(x)\frac{\partial}{\partial \theta}\log(p_\theta(x))p_\theta(x) \qquad \text{“log-likelihood trick”} \\
&= \mathbb{E}\left[f(X)\frac{\partial}{\partial \theta}\log(p_\theta(X))\right]
\end{aligned}$$

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

$$\begin{aligned}
\frac{\partial}{\partial\theta}\mathbb{E}[f(X)] &= \frac{\partial}{\partial\theta}\sum_{x\in\mathcal{X}} f(x)p_\theta(x) \\
&= \sum_{x\in\mathcal{X}} f(x)\frac{\partial}{\partial\theta}p_\theta(x) \\
&= \sum_{x\in\mathcal{X}} f(x)\frac{\partial}{\partial\theta}\log(p_\theta(x))p_\theta(x) \quad \text{“log-likelihood trick”} \\
&= \mathbb{E}\left[f(X)\frac{\partial}{\partial\theta}\log(p_\theta(X))\right]
\end{aligned}$$

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]

Decoder
D

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 $f : \mathcal{X} \to \mathbb{R}$

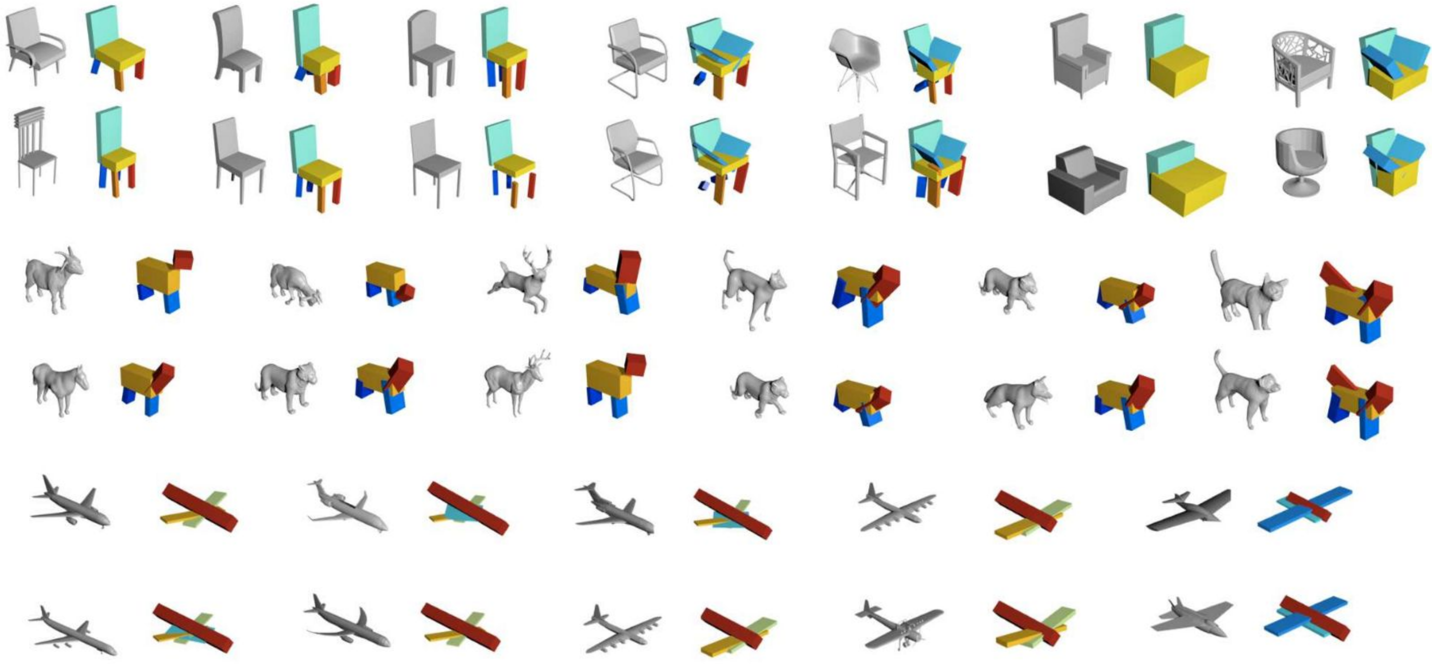
$$\frac{\partial}{\partial \theta}\mathbb{E}[f(X)] \approx \frac{1}{N}\sum_{n=1}^{N}\left[f(X_n)\frac{\partial}{\partial \theta}\log(p_\theta(X_n))\right]$$

Monte-Carlo sampling

This approximation is good when the dimension of $\mathcal{X}$ is not too high.

Learn 3D reconstruction with cuboids [Tulsiani2017]

Results

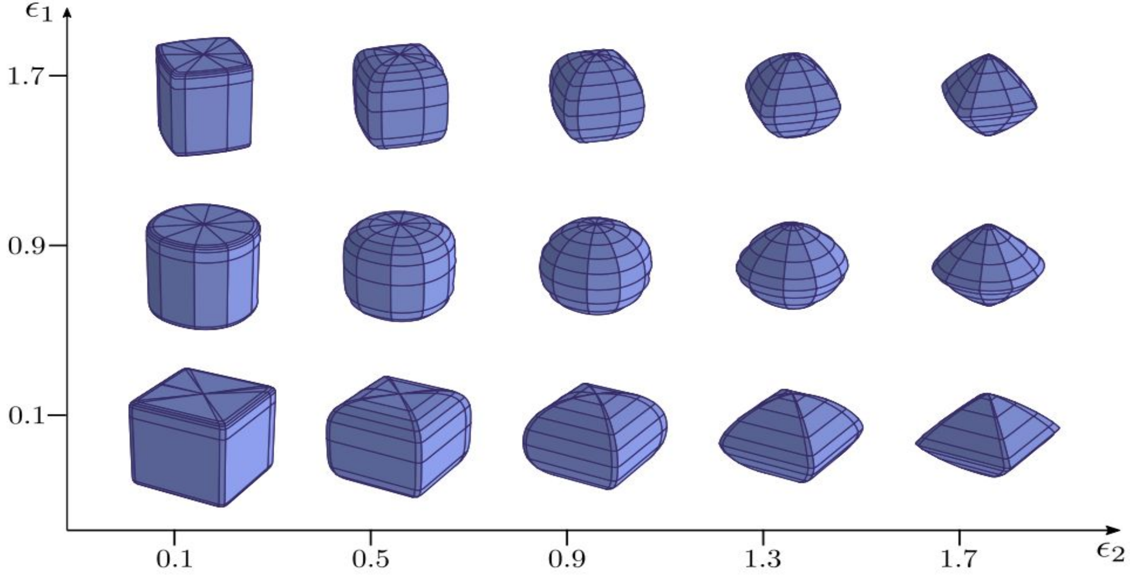


Notice that different shapes are reconstructed with different sets of cuboids

Learn 3D reconstruction with <u>superquadrics</u> [Paschalidou2019]

Decoder
D

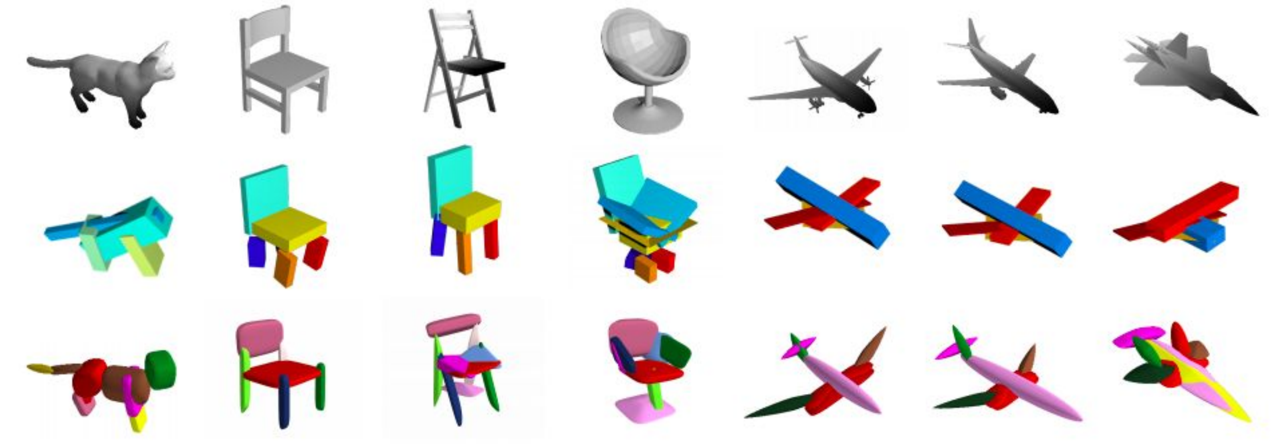
Everything in nature takes its form from the sphere, the cone and the cylinder.
- Paul Cezanne.



Superquadric Shape Space

Learn 3D reconstruction with superquadrics [Paschalidou2019]

Results



[Tulsiani2017]

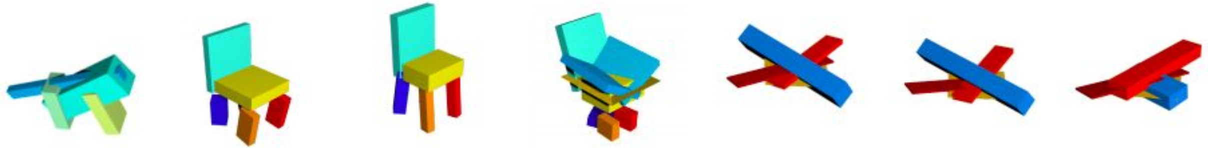
[Paschalidou2019]

Learn 3D reconstruction with superquadrics [Paschalidou2019]

Decoder
D

Results

[Tulsiani2017]



[Paschalidou2019]

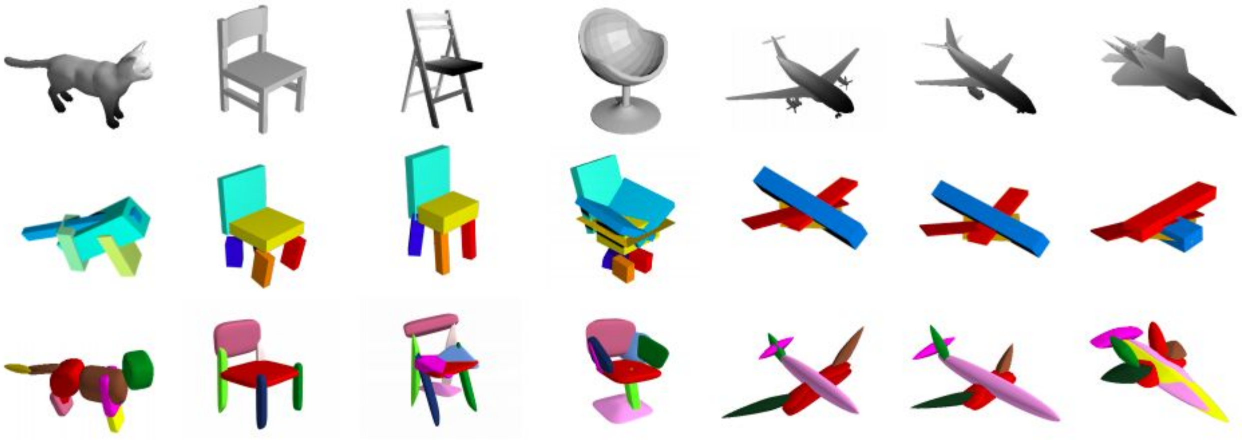
+ Generality
+ Parsimony of description
+ Inter-object coherence for signal transfer

- Data fidelity
- Training Stability

Learn 3D reconstruction with superquadrics [Paschalidou2019]

Decoder D

Results



[Tulsiani2017]

[Paschalidou2019]

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

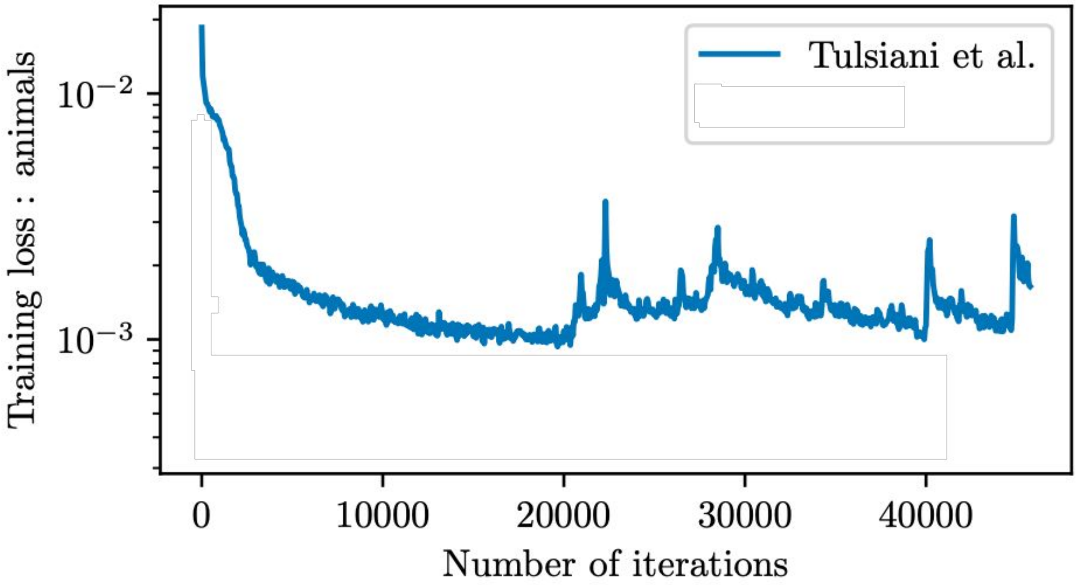
- Data fidelity
- Training Stability

Thibault Groueix, Pierre-Alain Langlois, 2019

Learn 3D reconstruction with superquadrics [Paschalidou2019]

Decoder D

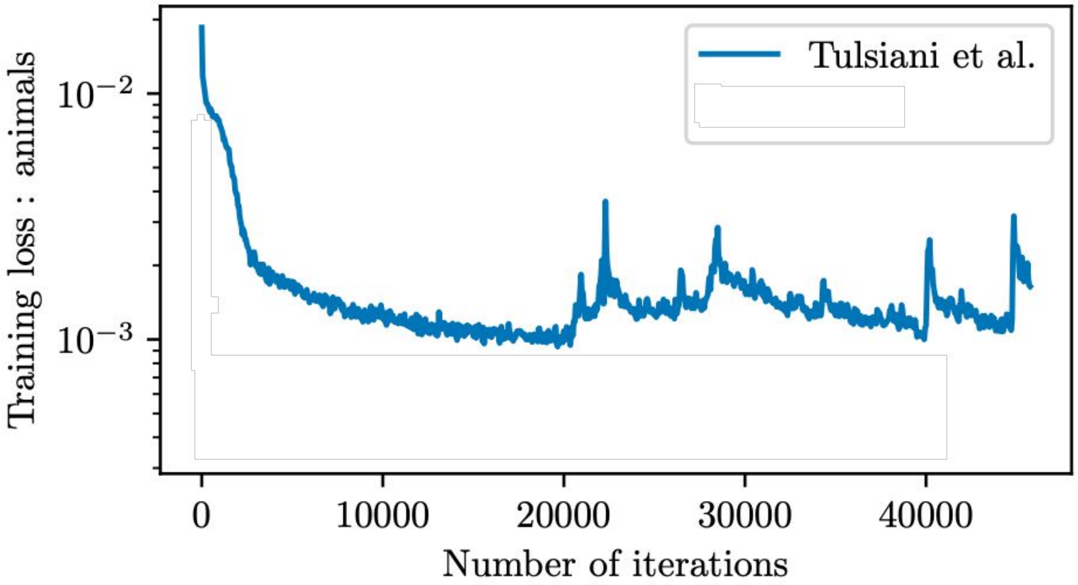
$$\frac{\partial}{\partial\theta}\mathbb{E}[f(X)] = \frac{\partial}{\partial\theta}\sum_{x\in\mathcal{X}} f(x)p_\theta(x)$$



(b) **Evolution of Training Loss.**

Learn 3D reconstruction with superquadrics [Paschalidou2019]

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



(b) **Evolution of Training Loss.**

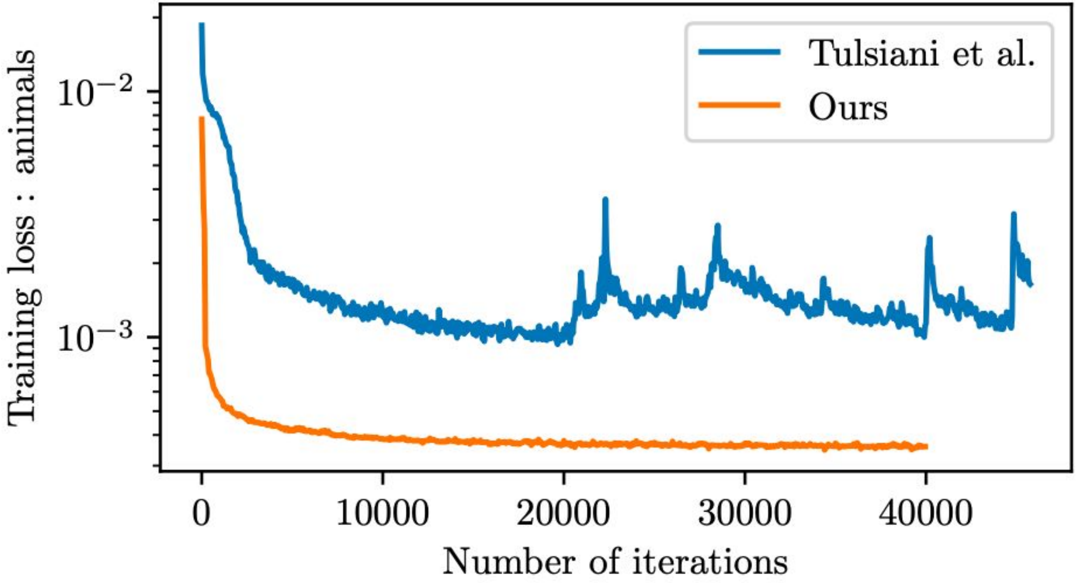
Are there alternatives to REINFORCE ?

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

Learn 3D reconstruction with superquadrics [Paschalidou2019]

Decoder D

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



(b) **Evolution of Training Loss.**

Are there alternatives to REINFORCE ?

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

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

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

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

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

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

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

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

$$\begin{aligned}
\mathbb{E}[f(X)] &= \mathbb{E}_{p(\theta)}\left[\left(\sum_{m=1}^{M}\mathcal{L}(\tilde{\mathbf{P}}_{\mathbf{m}},\mathbf{X})\right) + \mathcal{L}(\mathbf{X},\tilde{\mathbf{P}})\right] \\
&= \sum_{m=1}^{M}\theta_m\mathcal{L}_{(}\mathbf{P}_{\mathbf{m}},\mathbf{X}) + \mathbb{E}_{p(\theta)}\left[\sum_{\mathbf{x}_i\in\mathbf{X}}\min_{m|z_m=1}\Delta(\mathbf{x}_i,\mathbf{P}_{\mathbf{m}})\right] \\
&= \sum_{m=1}^{M}\theta_m\mathcal{L}_{(}\mathbf{P}_{\mathbf{m}},\mathbf{X}) + \mathbb{E}_{p(\theta)}\left[\sum_{\mathbf{x}_i\in\mathbf{X}}\min_{m|z_m=1}\Delta_i^m,\mathbf{P}_{\mathbf{m}})\right]
\end{aligned}$$

2^M configurations

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

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

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

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

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

2^M configurations

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

$$\Delta_i^1 \leq \Delta_i^2 \leq \cdots \leq \Delta_i^M$$

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

Reordering trick

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

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

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

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

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

2^M configurations

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

$$\Delta_i^1 \leq \Delta_i^2 \leq \cdots \leq \Delta_i^M$$

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

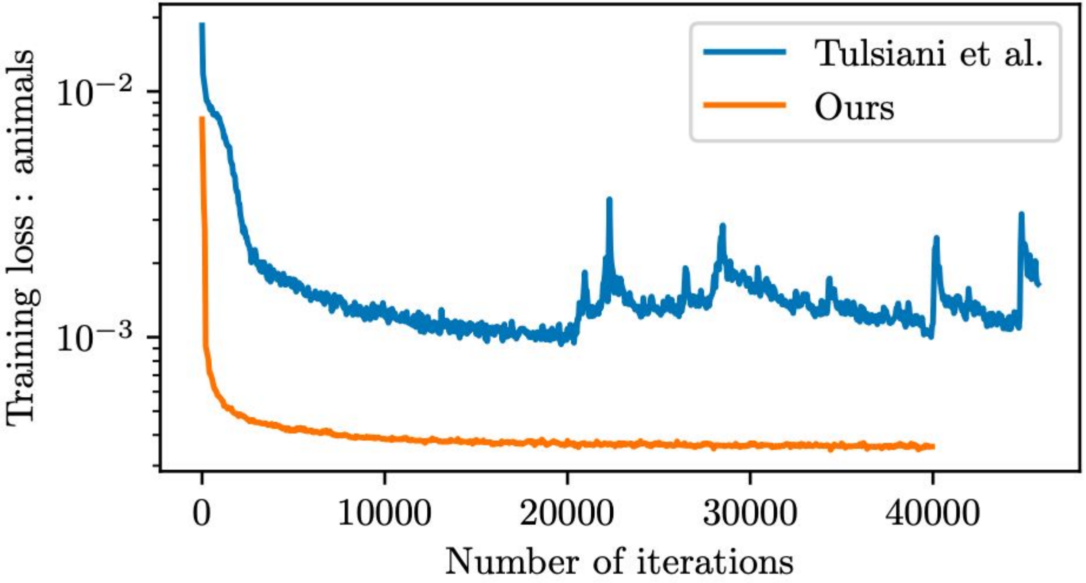
Reordering trick

$$\boxed{\mathbb{E}[f(X)] = \sum_{m=1}^{M} \theta_m \mathcal{L}_(\mathbf{P}_\mathbf{m},\mathbf{X}) + \sum_{\mathbf{x}_i \in \mathbf{X}} \sum_{m=1}^{M} \Delta_i^m \theta_m \prod_{\overline{m}=1}^{m-1} (1 - \theta_{\overline{m}})}$$

Complexity: 2^M -> M^2

Learn 3D reconstruction with superquadrics [Paschalidou2019]

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

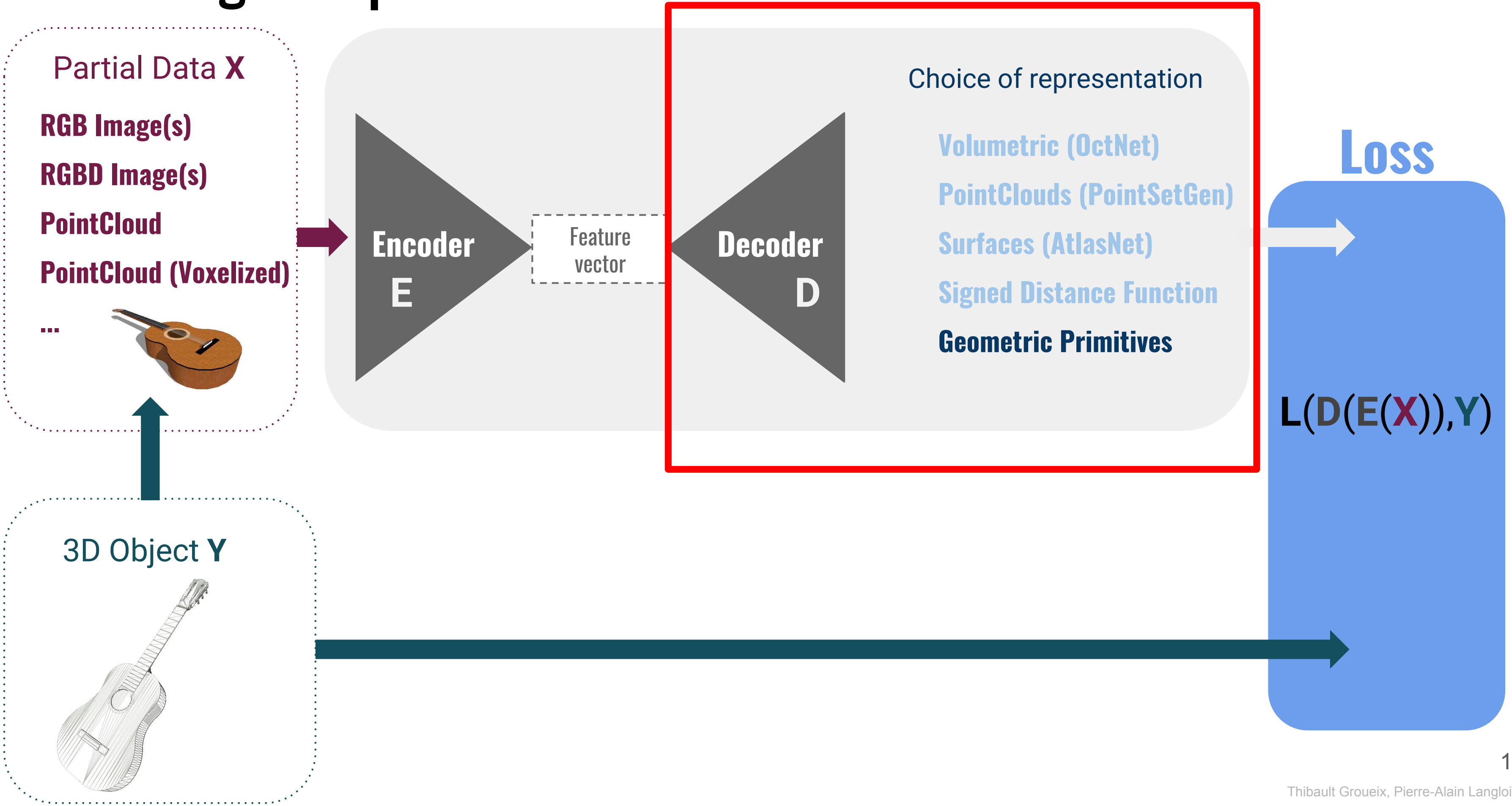


(b) **Evolution of Training Loss.**

Are there alternatives to REINFORCE ?

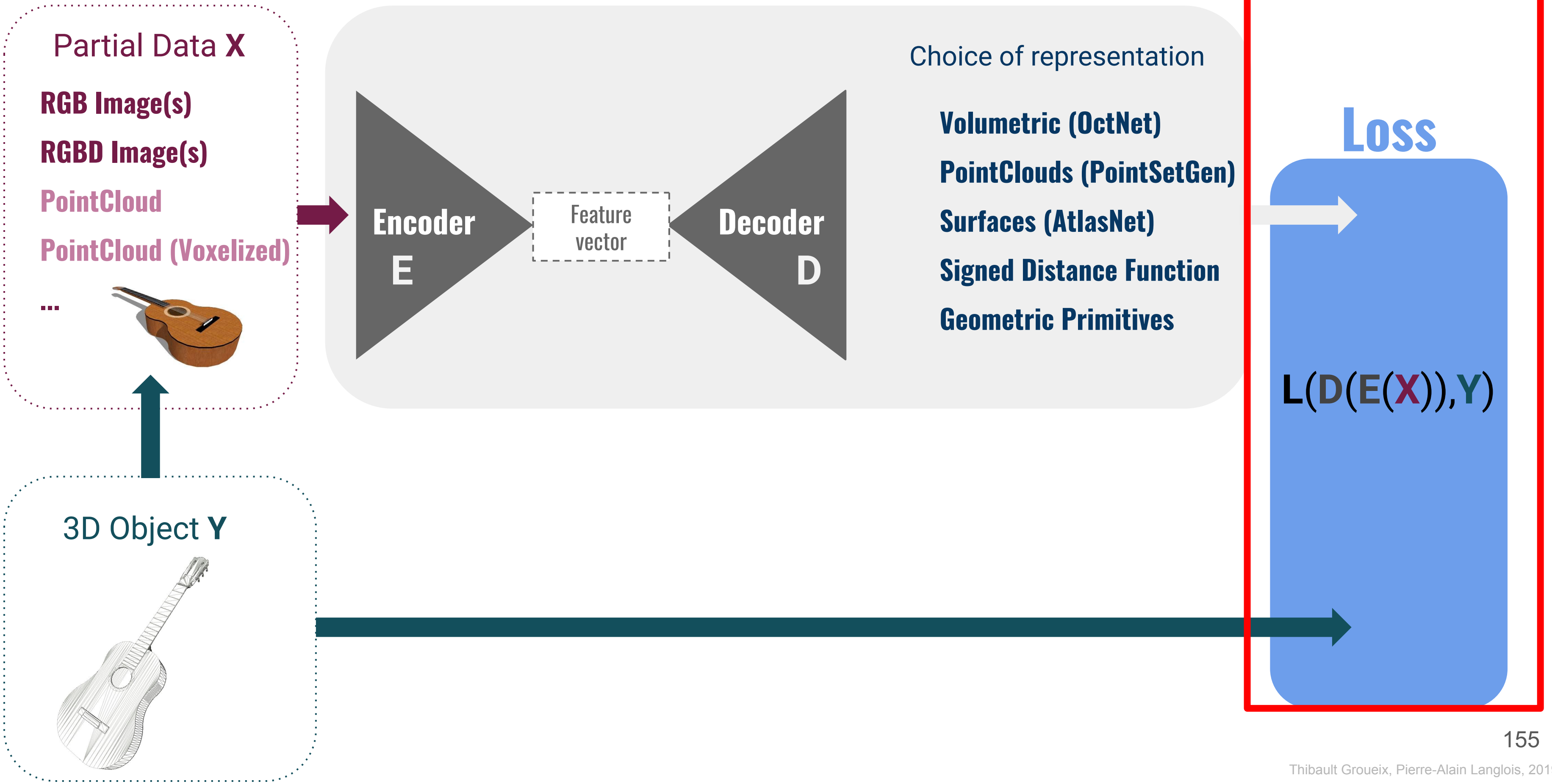
- **The Reparameterization trick : cf [MohamedSlides]**
- **Direct analytical computation in the particular case of the Chamfer Distance**

Training setup for 3D reconstruction



Thibault Groueix, Pierre-Alain Langlois, 2019

Training setup for 3D reconstruction



Thibault Groueix, Pierre-Alain Langlois, 2019

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

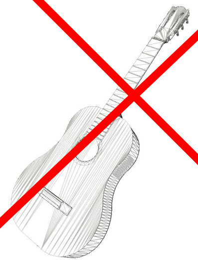
Geometric Primitives

Loss

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

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

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

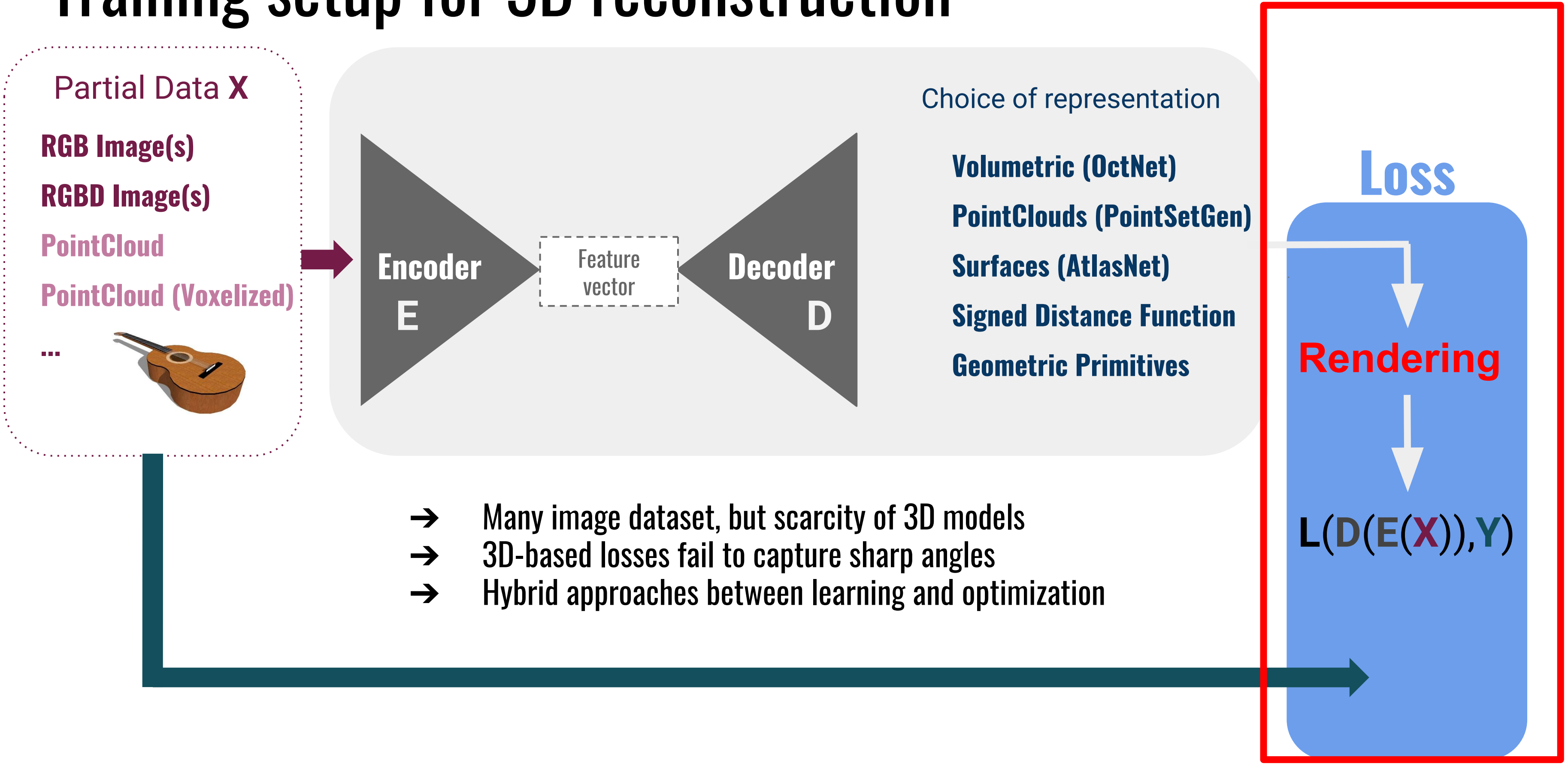
Geometric Primitives

Loss

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

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

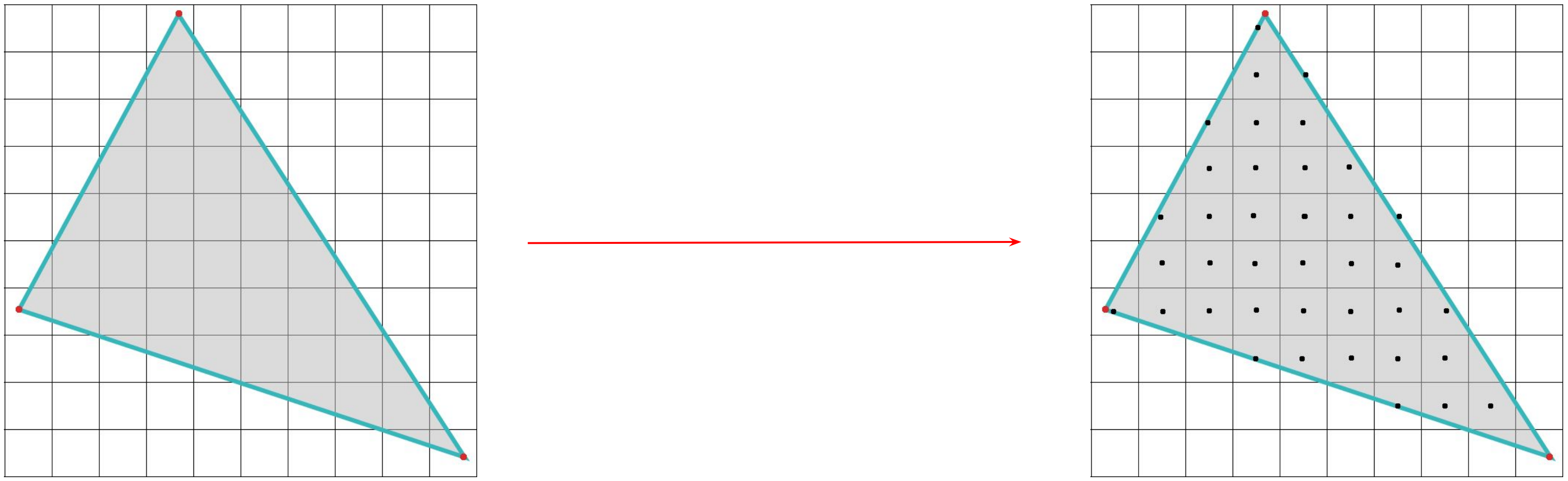
Training setup for 3D reconstruction



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

Inverse Rendering - Issues

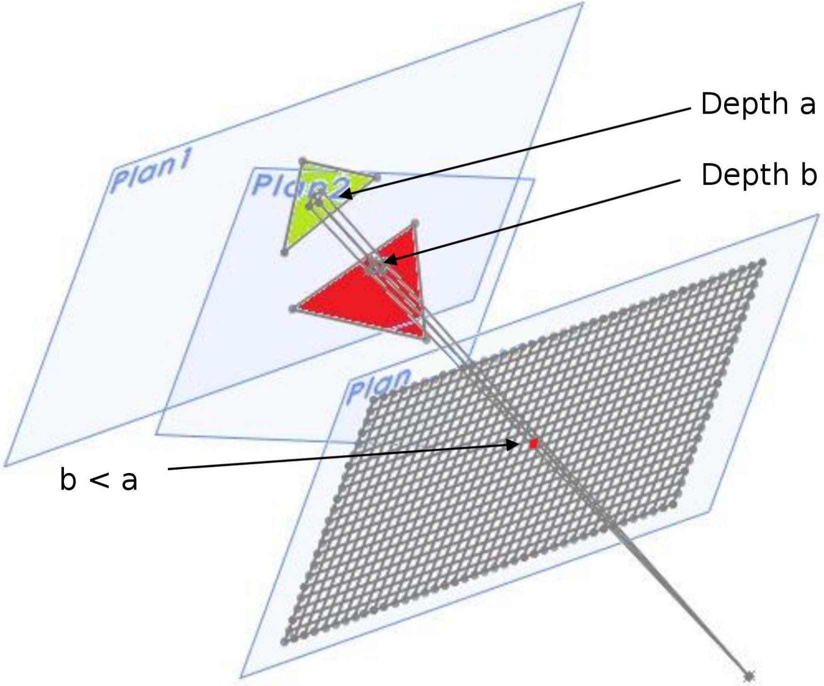
➔ Rasterization is not differentiable



Thibault Groueix, Pierre-Alain Langlois, 2019

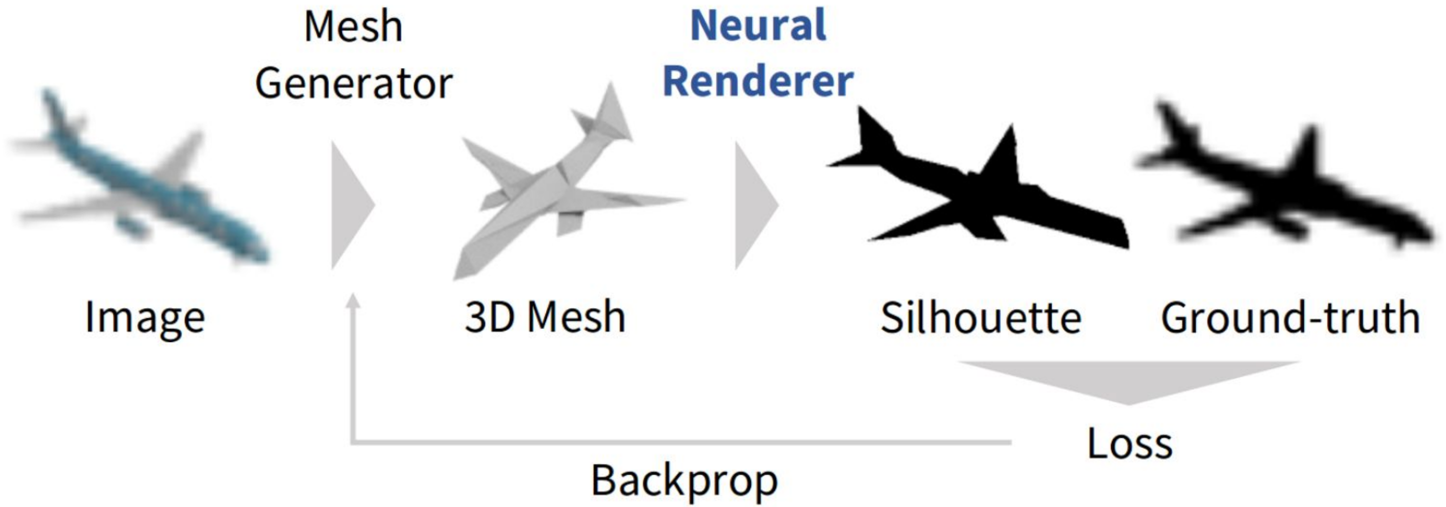
Inverse Rendering - Issues

➔ Z-buffering is not differentiable



Differentiating the rasterization process - silhouette case
[Kato2018]

➔ Avoid the z-buffering process by just rendering sillouhettes



Differentiating the rasterization process - silhouette case
[Kato2018]

➔ Get a differentiable process through blurring

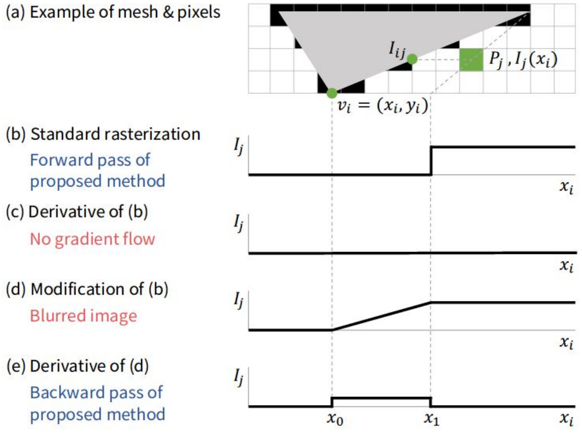


Figure 2. Illustration of our method. $\boldsymbol{v}_i = \{x_i, y_i\}$ is one vertex of the face. I_j is the color of pixel P_j. The current position of x_i is x_0. x_1 is the location of x_i where an edge of the face collides with the center of P_j when x_i moves to the right. I_j becomes I_{ij} when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

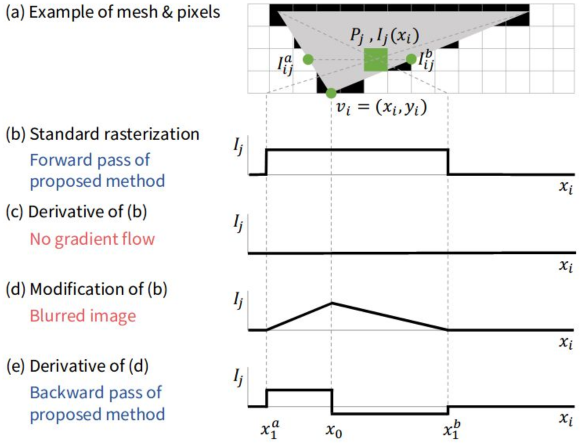


Figure 3. Illustration of our method in the case where P_j is inside the face. I_j changes when x_i moves to the right or left.

Differentiating the rasterization process - silhouette case

[Kato2018]

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

Differentiating the rasterization process - silhouette case
[Kato2018]

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

Problem:

➔ Requires strong regularization to work !

Figure 5. Generation of the back side of a CRT monitor with/without smoothness regularizer. Left: input image. Center: prediction without regularizer. Right: prediction with regularizer.

Differentiating the rasterization process - silhouette case

[Kato2018]

Problem:

➔ Requires strong regularization to work !

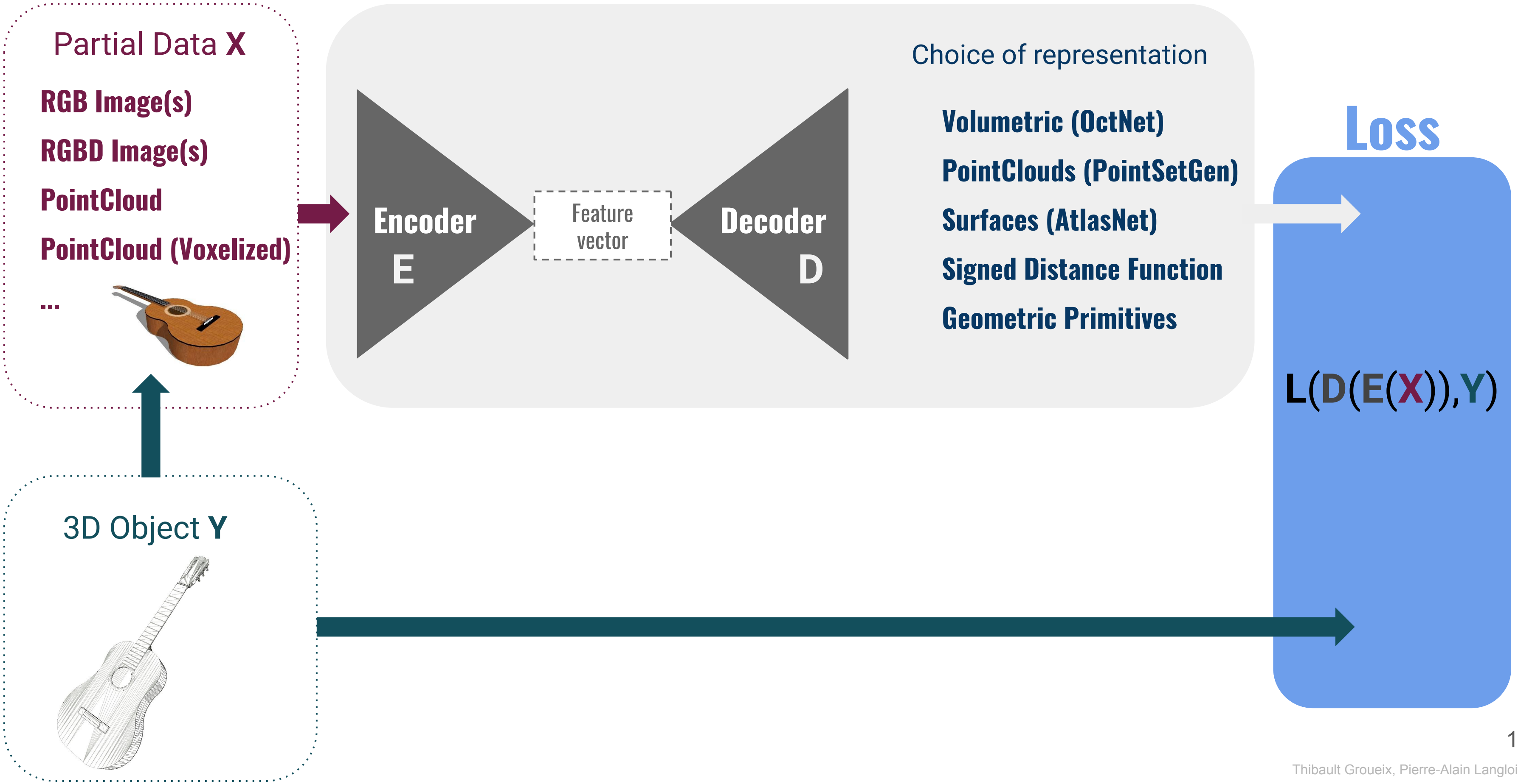
Ideas:

➔ Use multiple views **[Petersen2019]**

➔ Improve the rendering process
[Nguyen-Phuoc2018], [Petersen2019], [Yang2018]

Thibault Groueix, Pierre-Alain Langlois, 2019

Training setup for 3D reconstruction



Very Modular Framework ! [Tulsiani2017]

Partial Data **X**

RGB Image(s)

RGBD Image(s)

PointCloud

PointCloud (Voxelized)

...

3D Object **Y**

Encoder
E

Feature vector

Decoder
D

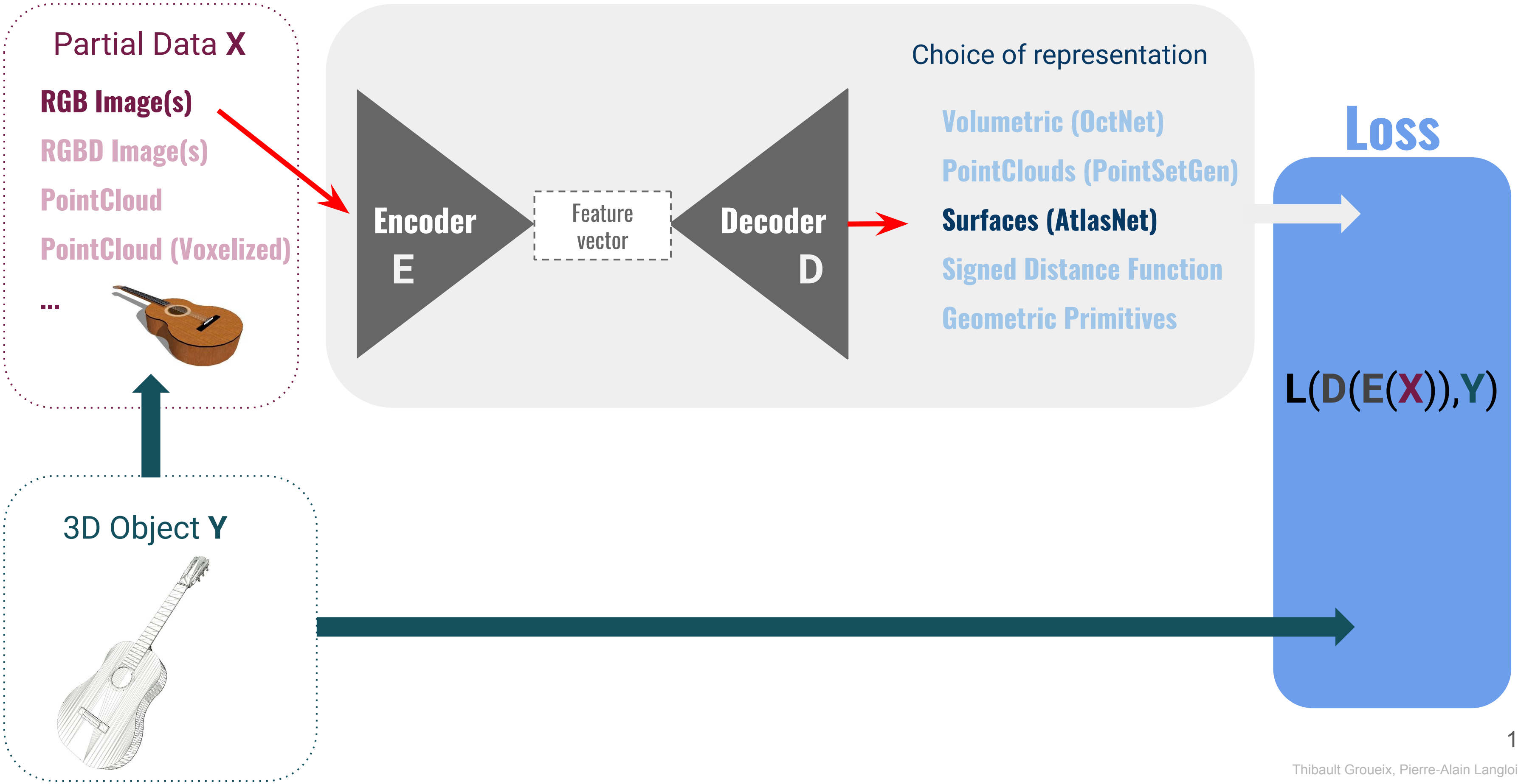
Choice of representation

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

Loss

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

Very Modular Framework ! [Groueix2018]



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
- ➔ ...

Thibault Groueix, Pierre-Alain Langlois, 2019

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.
- ★ **[Paschalidou2019]** : Paschalidou, Despoina, Ali Osman Ulusoy, and Andreas Geiger. "Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids." 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019.

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
- **[Kato2018]** : Kato, Hiroharu, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3d mesh renderer." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3907-3916. 2018
- **[Nguyen-Phuoc2018]** : Nguyen-Phuoc, Thu H., Chuan Li, Stephen Balaban, and Yongliang Yang. "RenderNet: A deep convolutional network for differentiable rendering from 3D shapes." In Advances in Neural Information Processing Systems, pp. 7891-7901. 2018.
- **[Petersen2019]** : Petersen, Felix, Amit H. Bermano, Oliver Deussen, and Daniel Cohen-Or. "Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer." arXiv preprint arXiv:1903.11149 (2019).
- **[Yang2018]** : Yang, Dawei, Chaowei Xiao, Bo Li, Jia Deng, and Mingyan Liu. "Realistic adversarial examples in 3d meshes." arXiv preprint arXiv:1810.05206 (2018).
- **[MohamedSlides] :** Shakir Mohamed https://www.shakirm.com/slides/MLSS2018-Madrid-ProbThinking.pdf