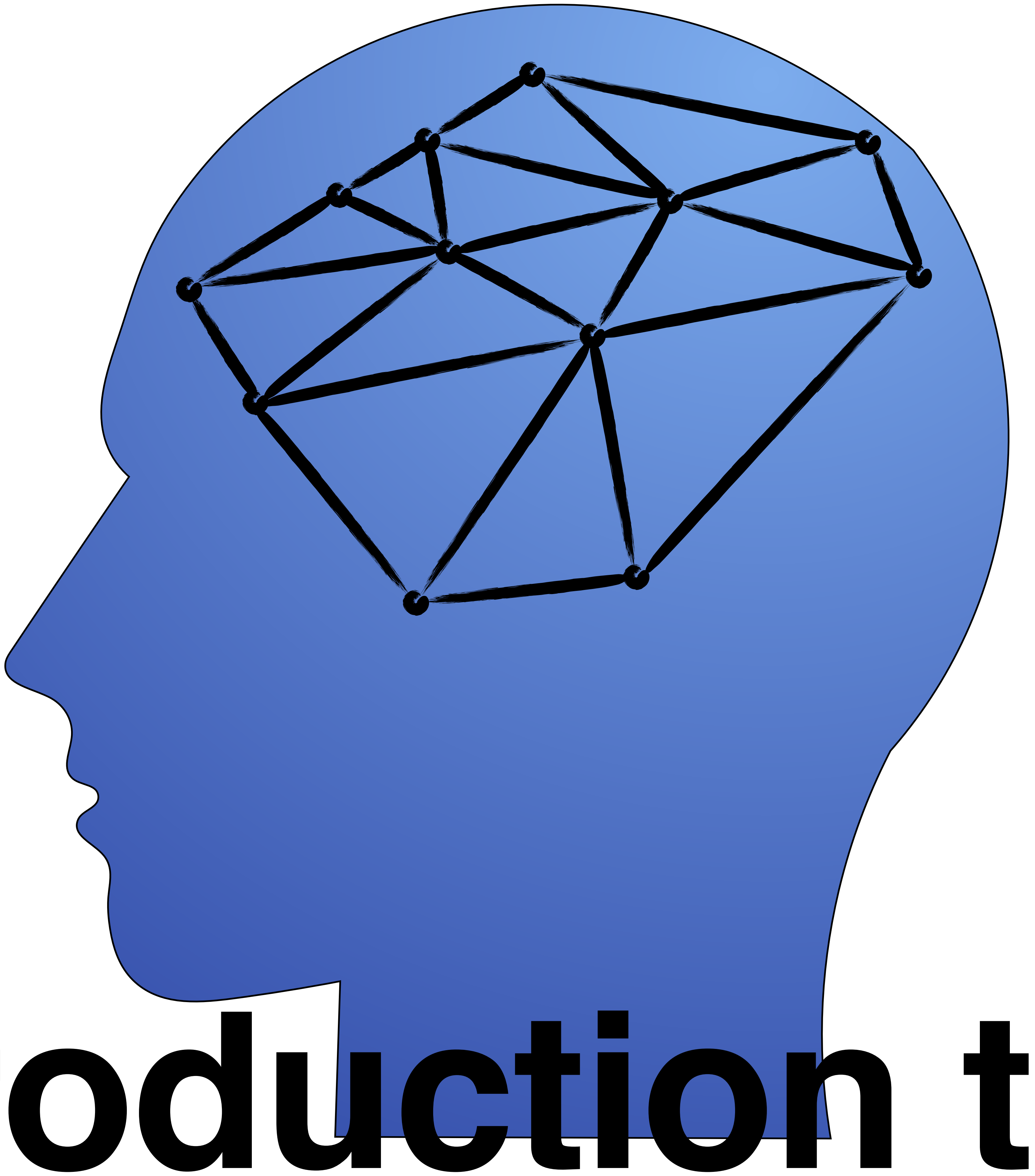


An Introduction to Deep Learning on Meshes

SIGGRAPH COURSE 2021

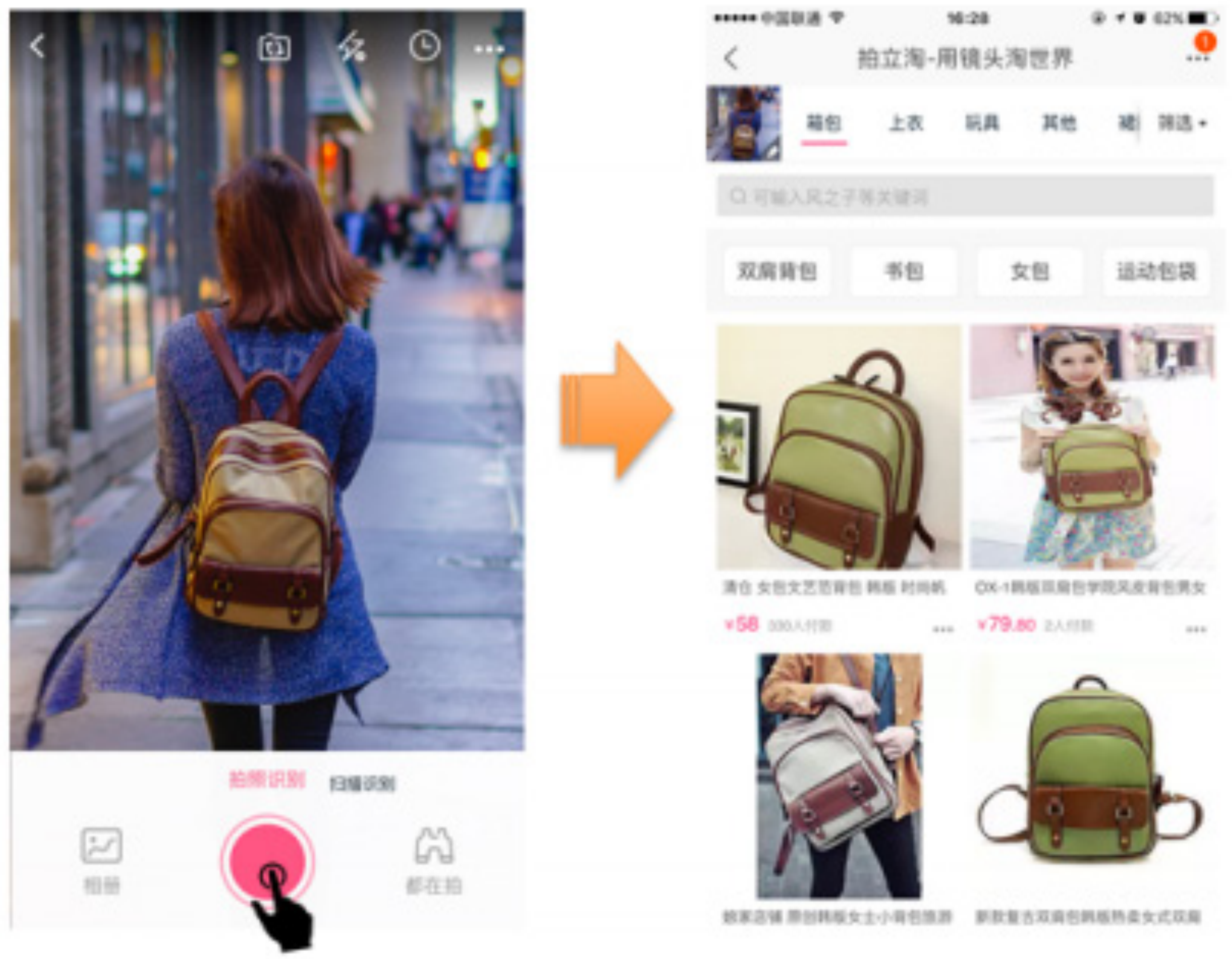
Rana Hanocka & Hsueh-Ti Derek Liu

THE UNIVERSITY OF CHICAGO

UNIVERSITY OF TORONTO

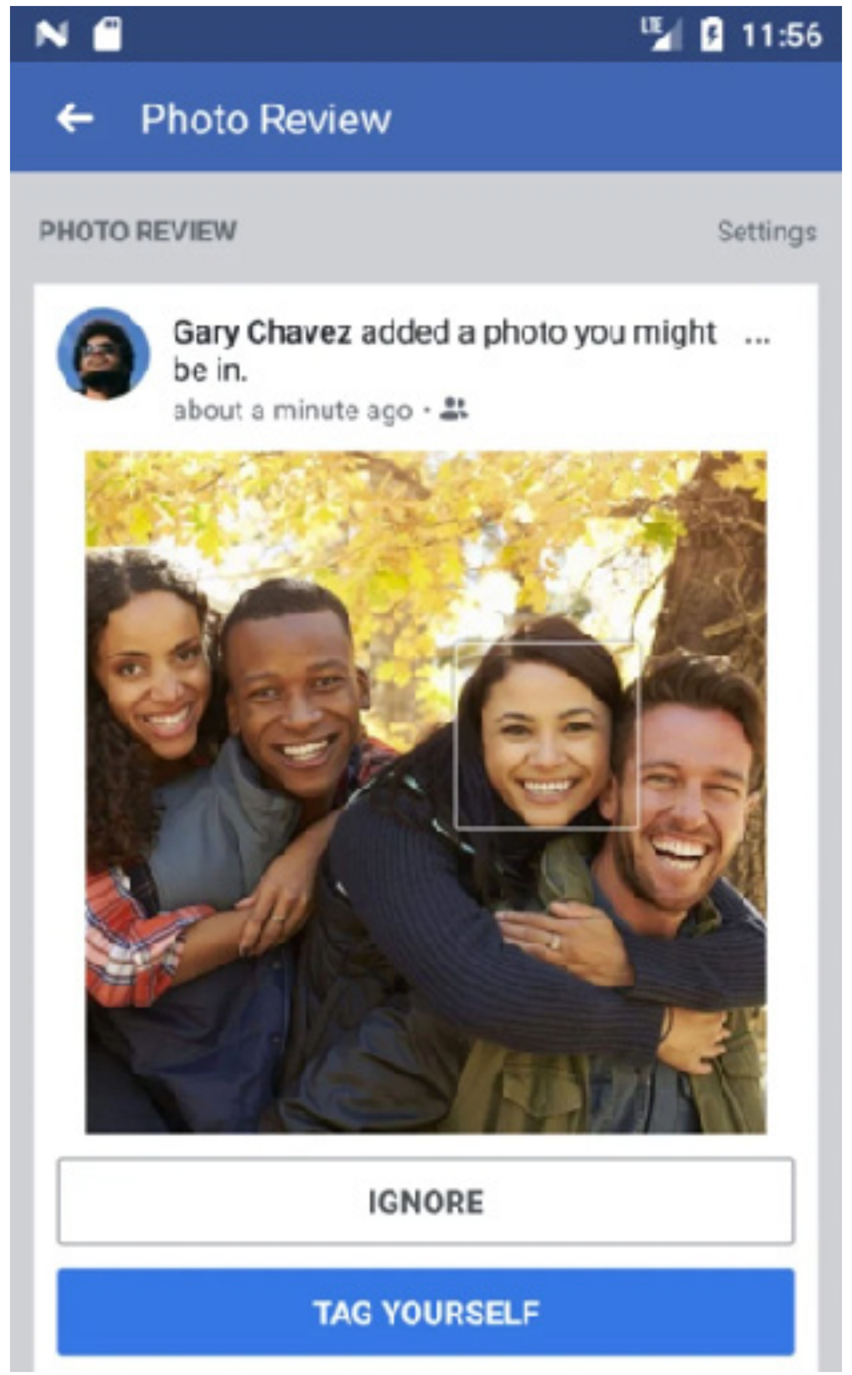
Real world success of deep learning

Reverse image search



Alibaba Pailitao

Facial recognition



Gary Chavez added a photo you might be in.

about a minute ago ·

Facebook Photo Tags

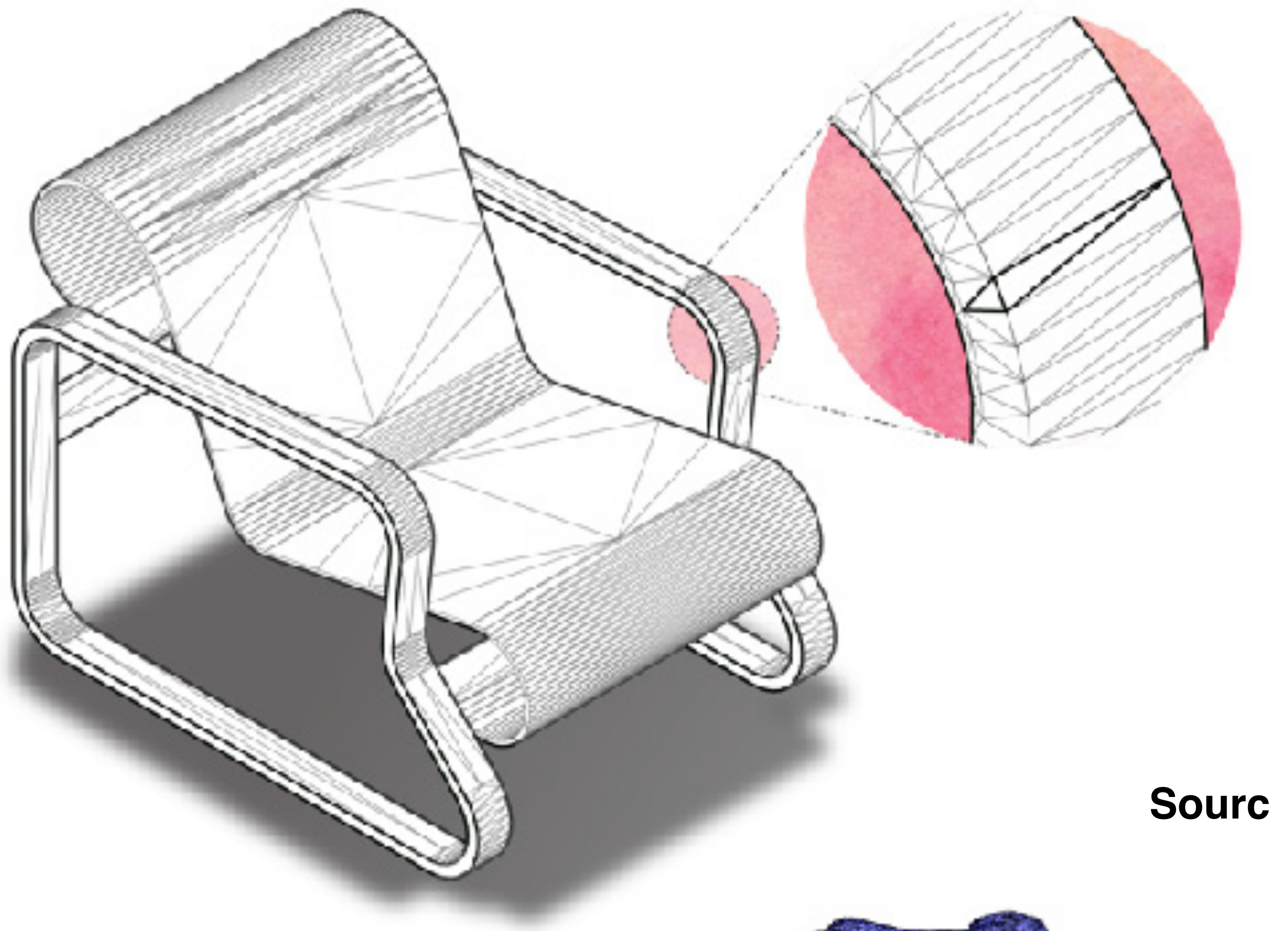
Speech recognition /
Language processing

Apple Siri

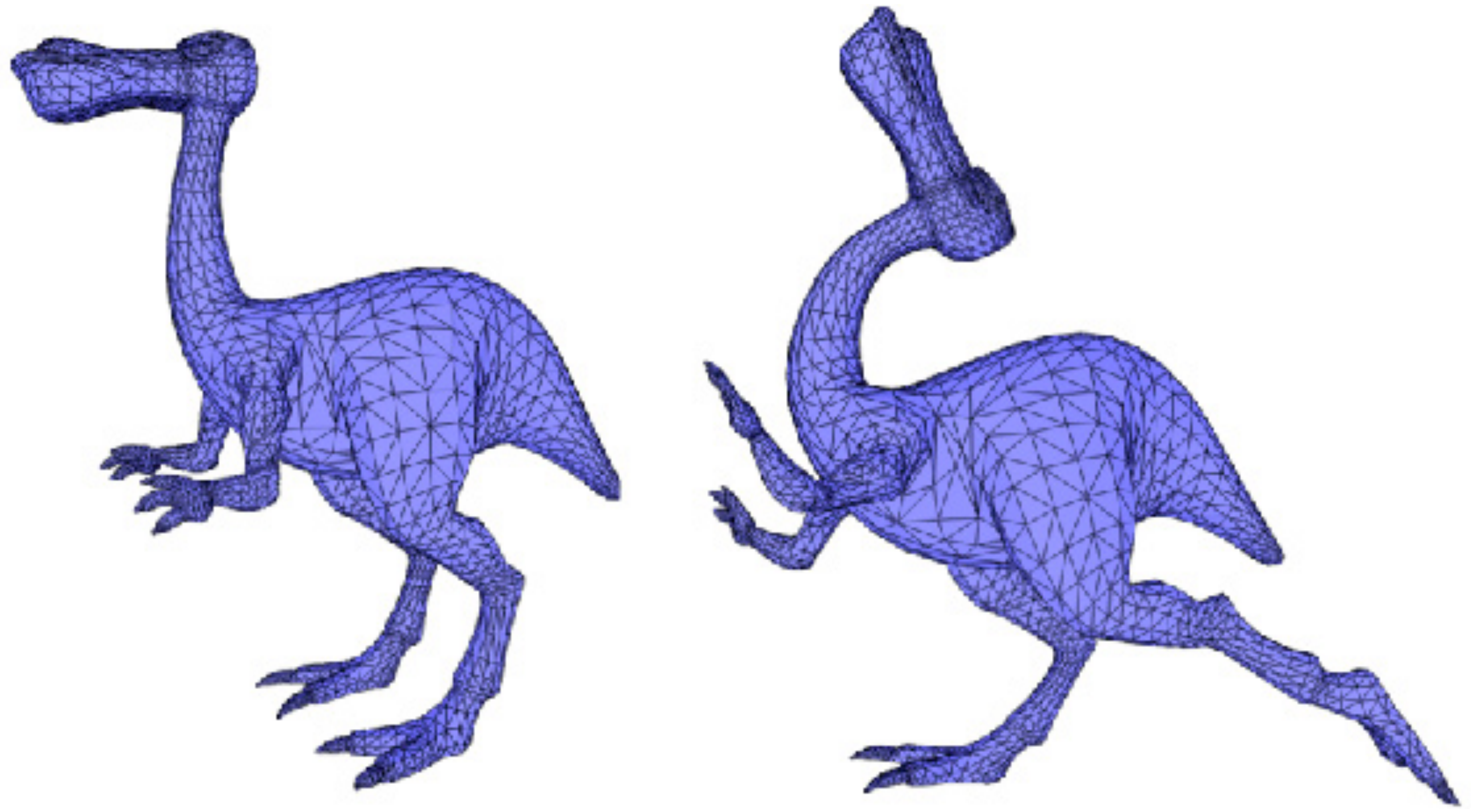
machine translation

Google translate

Meshes are popular in computer graphics



Source: Sorkine & Alexa 2007



Source: Sawhney & Crane 2017

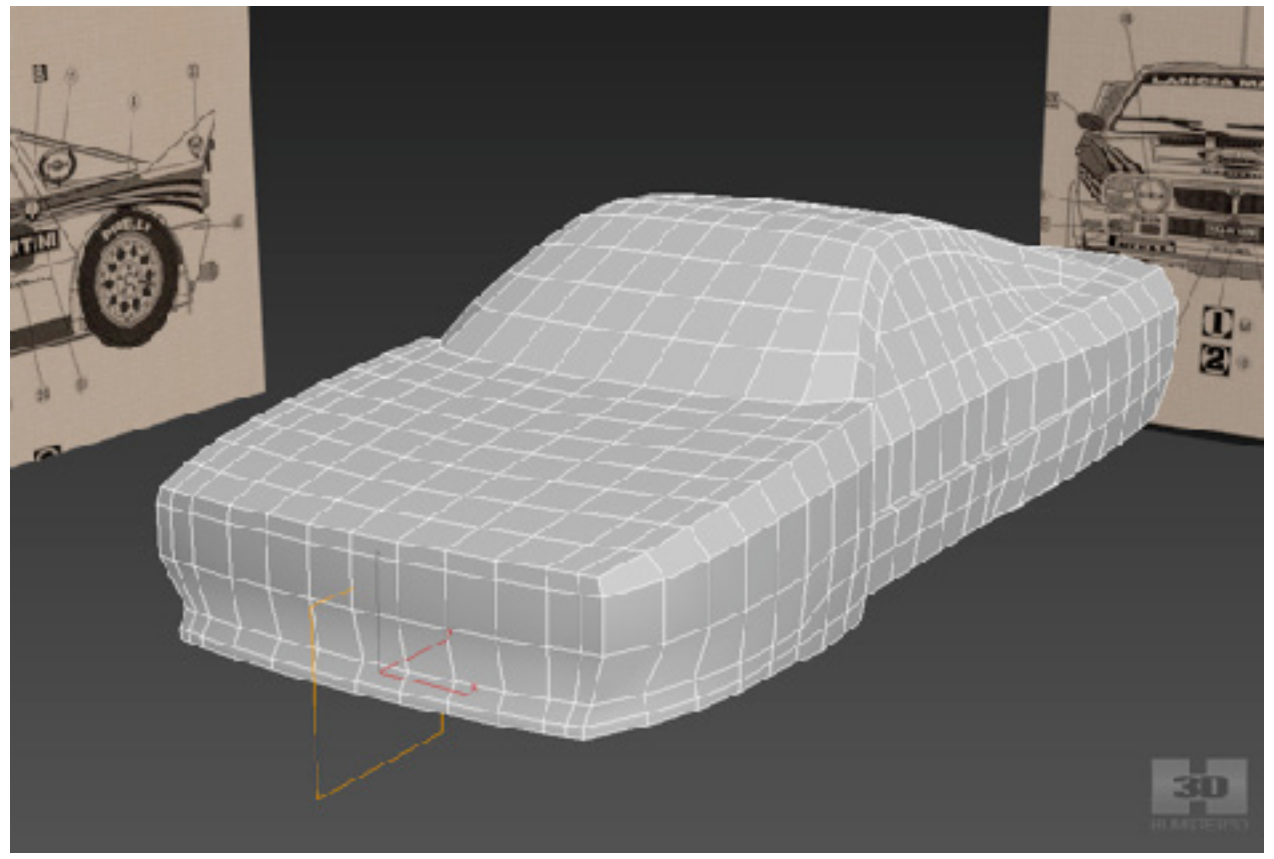
Source: Li et. al 2020

fast to render adaptive efficient to texture intuitively deformable physics simulation

Combining the power of deep learning & meshes

for many applications in geometry processing

Modeling



Editing

Reconstruction

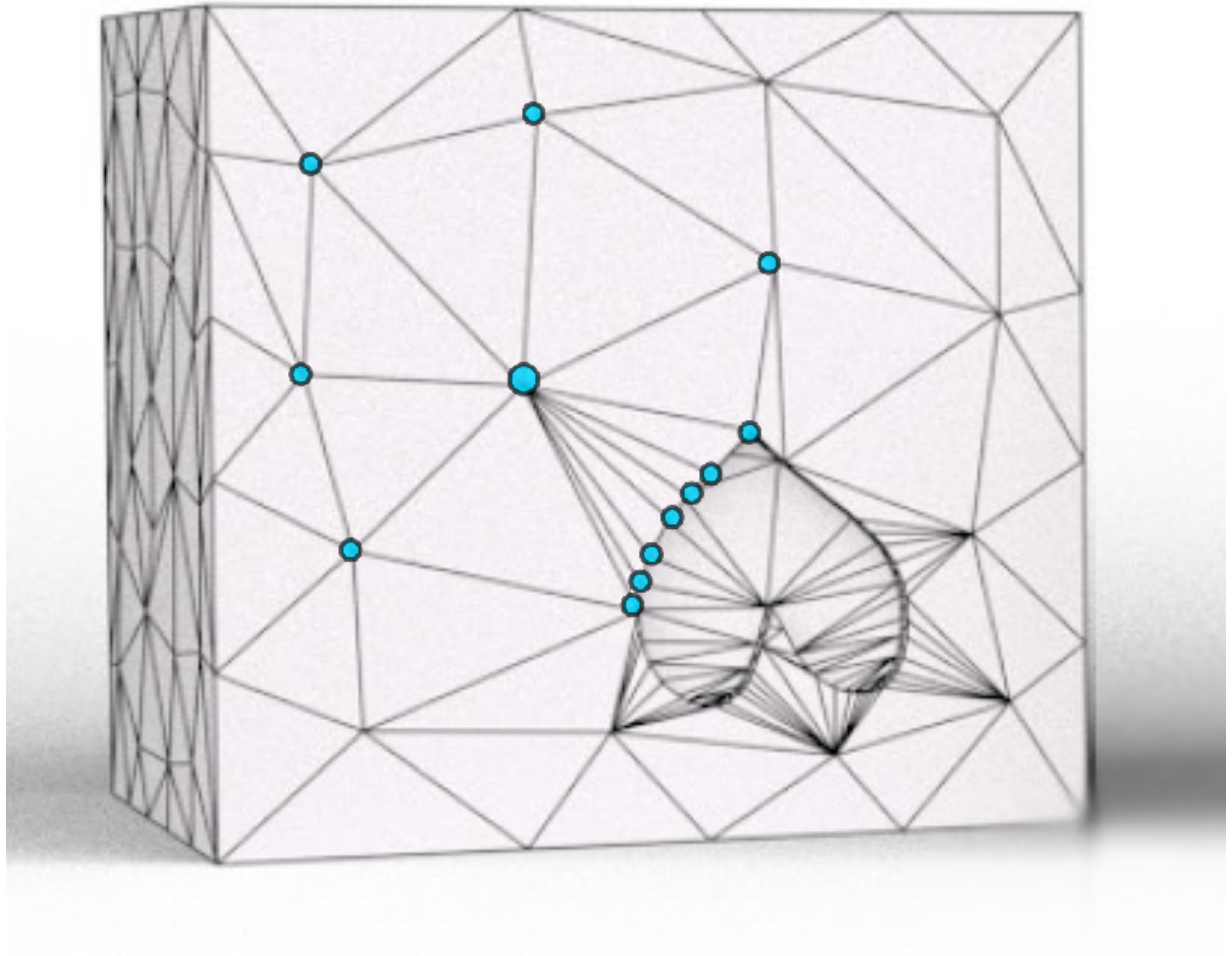
Shape Analysis

Challenges for deep learning on meshes

Representation

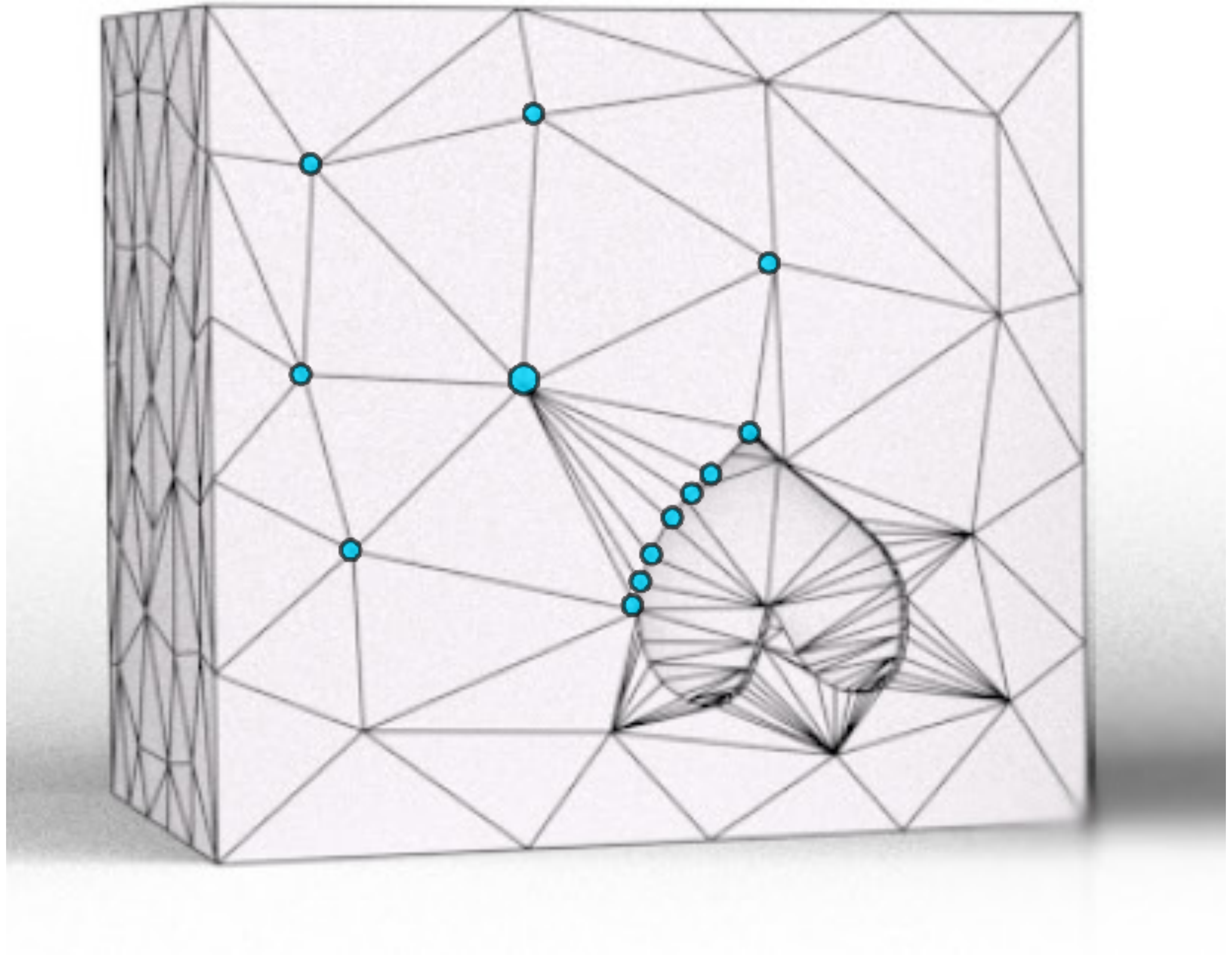
Data Accessibility

Irregular

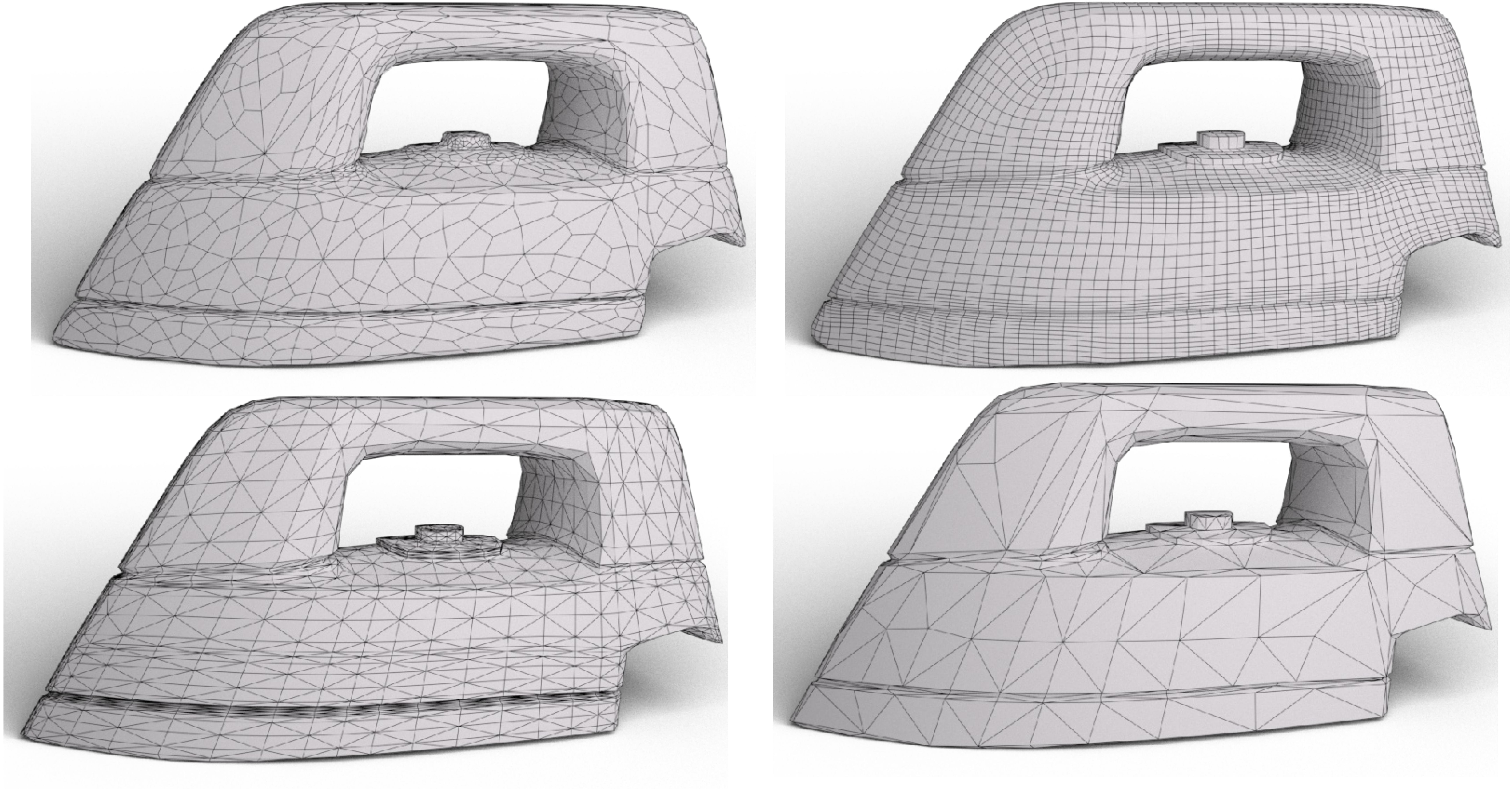


Unordered

<x,	y,	z>
<1.2,	3.1,	-0.7>
<1.2,	3.4,	-0.8>
<1.5,	3.4,	-0.6>
<1.5,	3.1,	-0.7>
<1.5,	3.7,	-0.7>
<1.7,	3.4,	-0.7>



Inconsistent



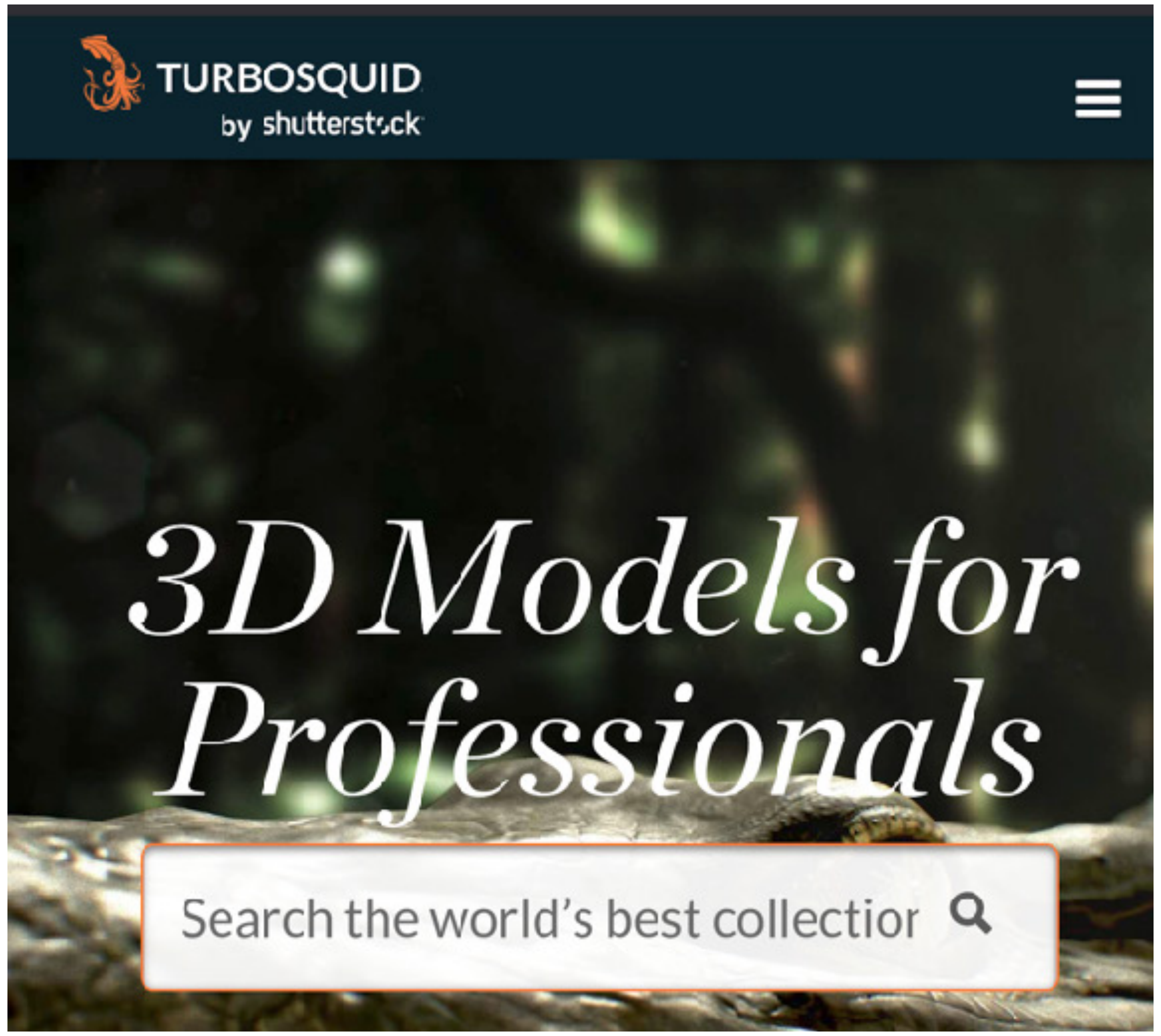
Unoriented

Challenges for deep learning on meshes

Representation

Data Accessibility

Large Warehouses of 3D Mesh Data



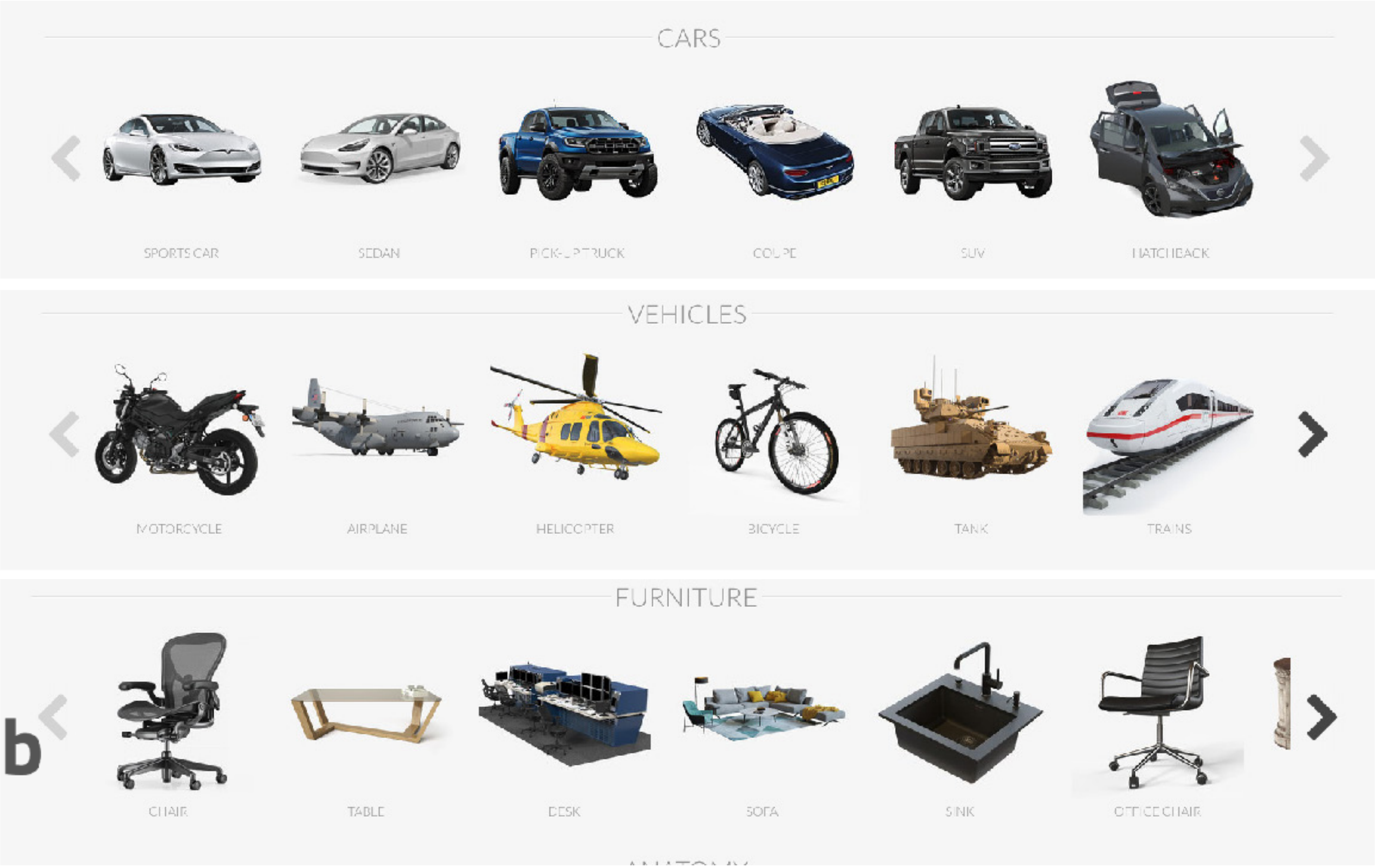
TURBOSQUID
by shutterstock

3D Models for Professionals

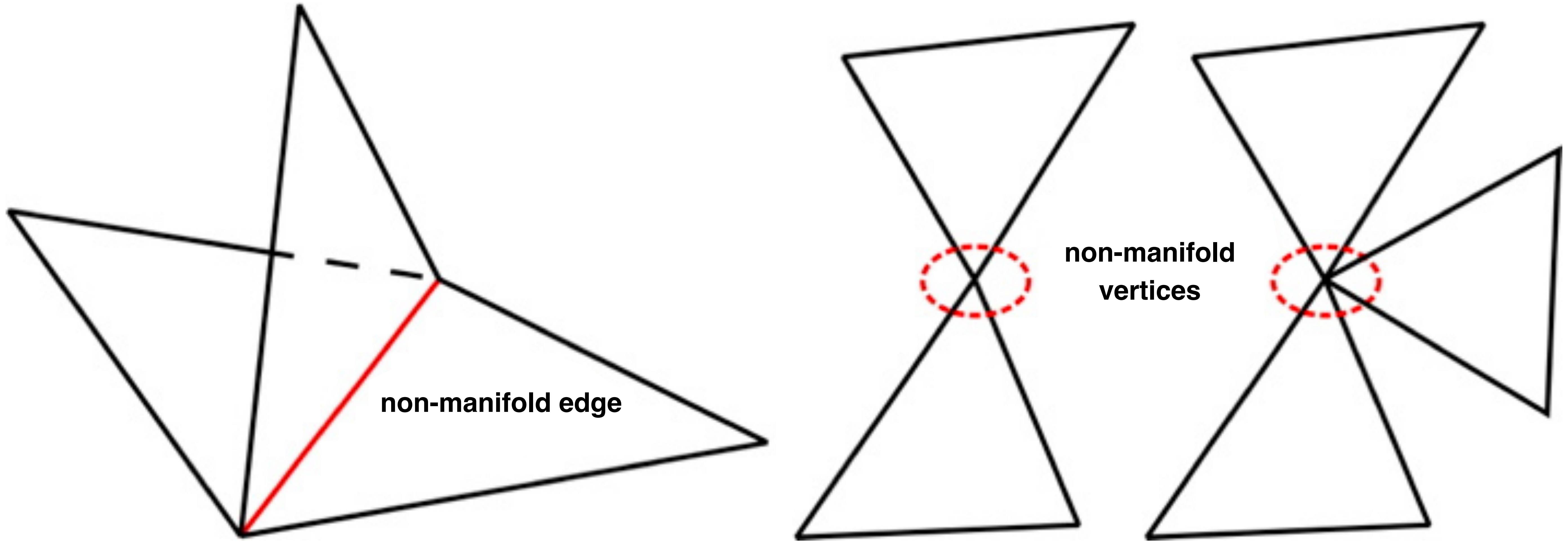
Search the world's best collection

3D Warehouse

Sketchfab

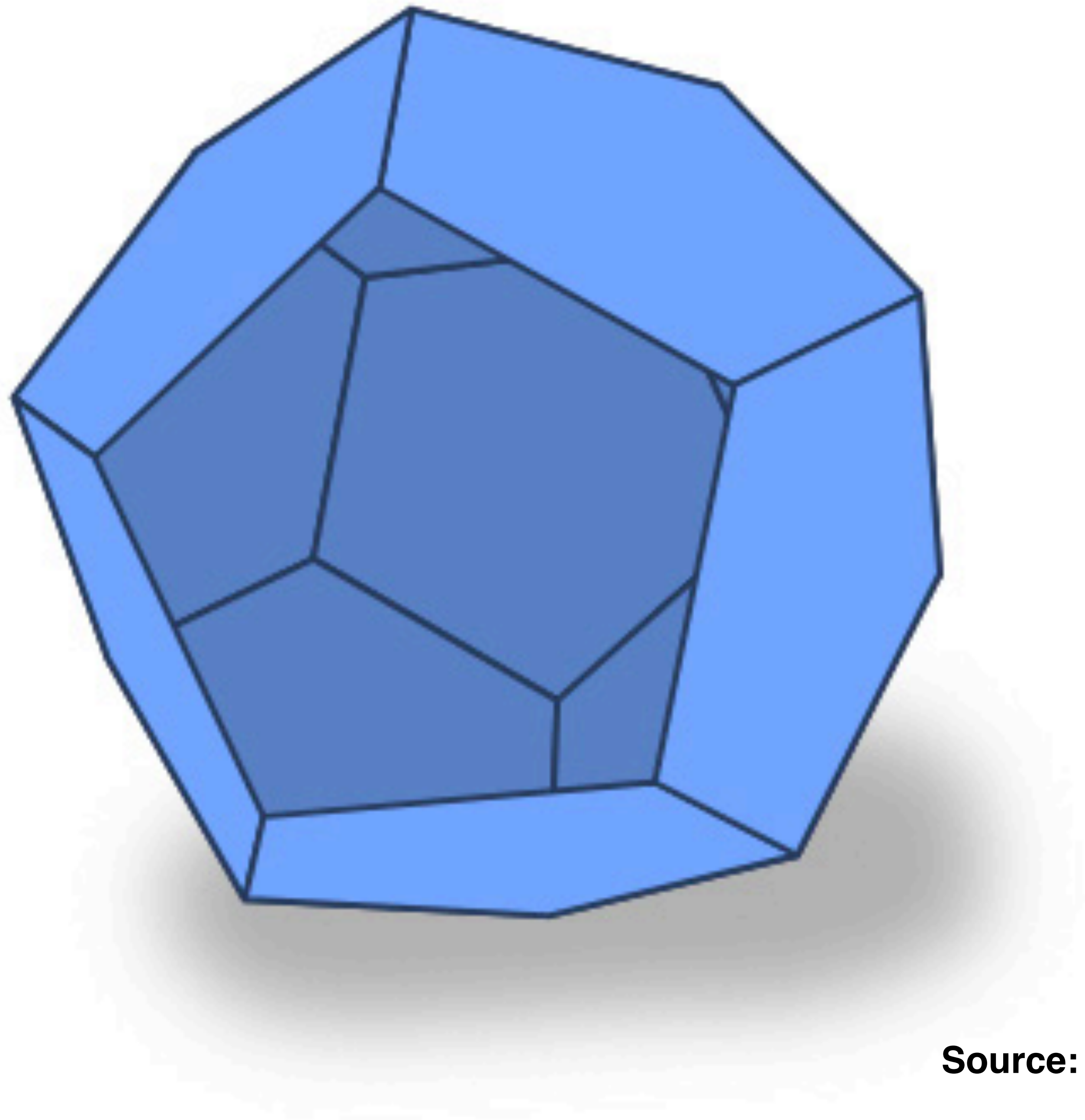


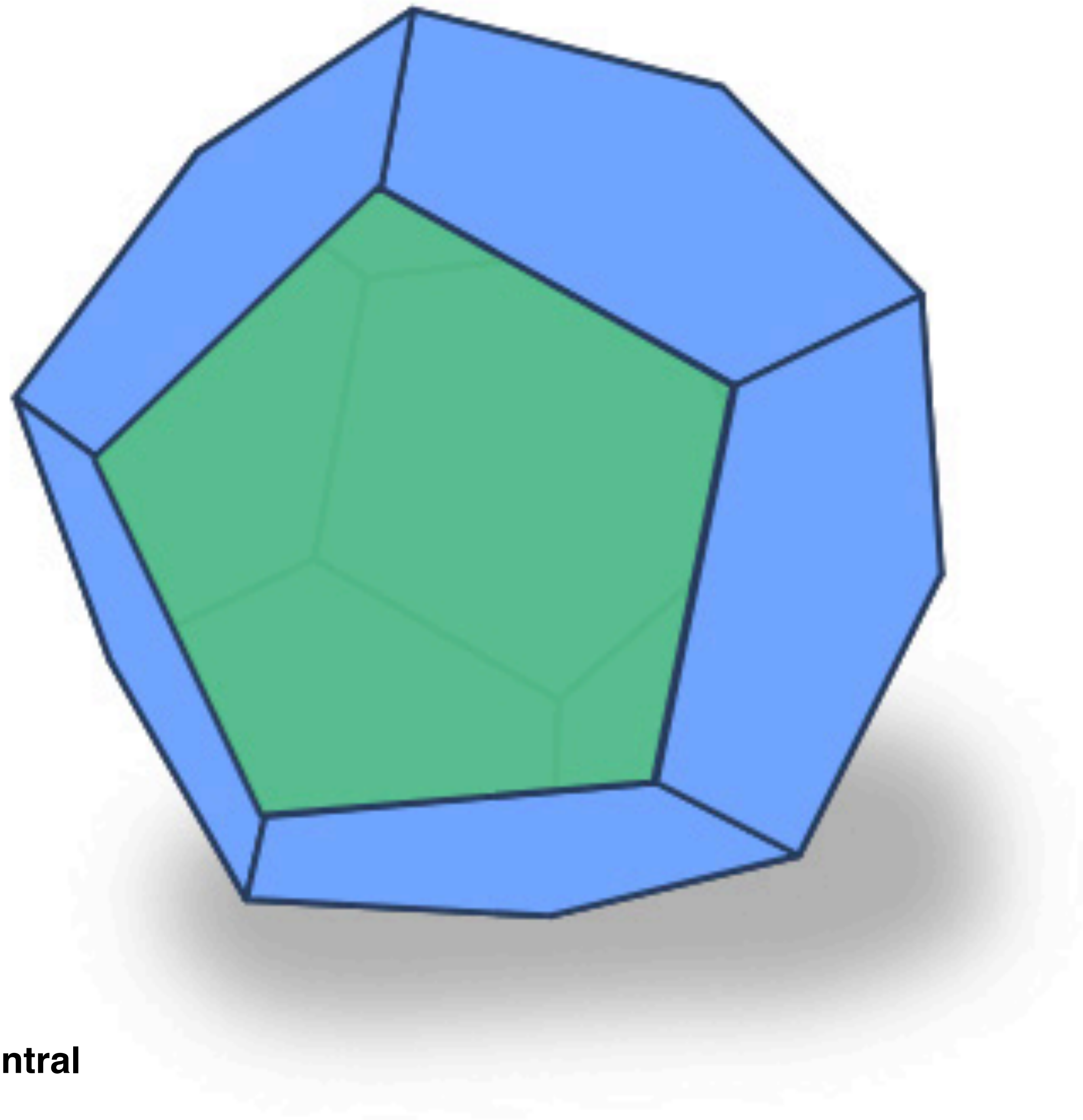
High bar for geometric computation



non-manifold

High bar for geometric computation

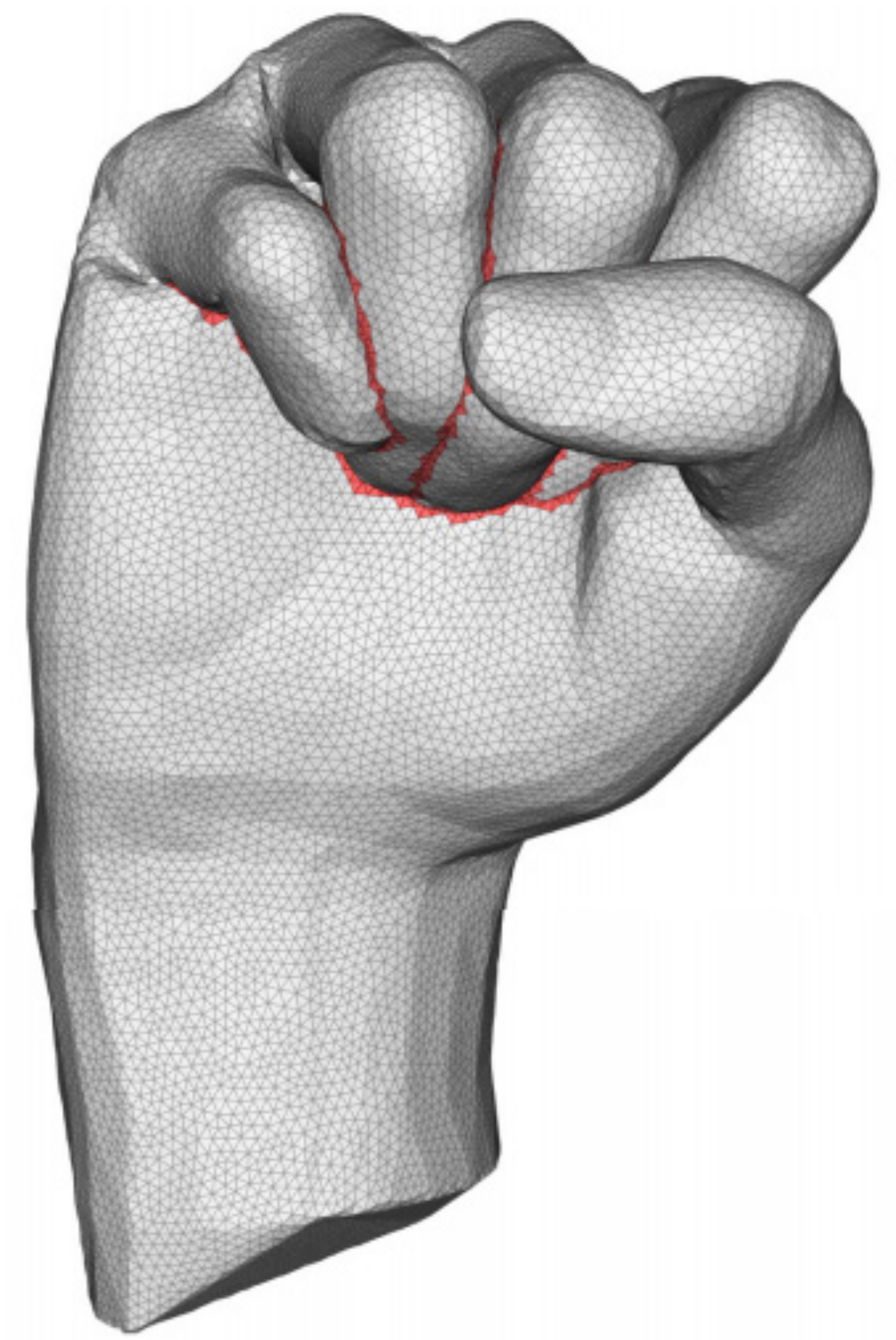




Source: geometry central

non-manifold not watertight

High bar for geometric computation

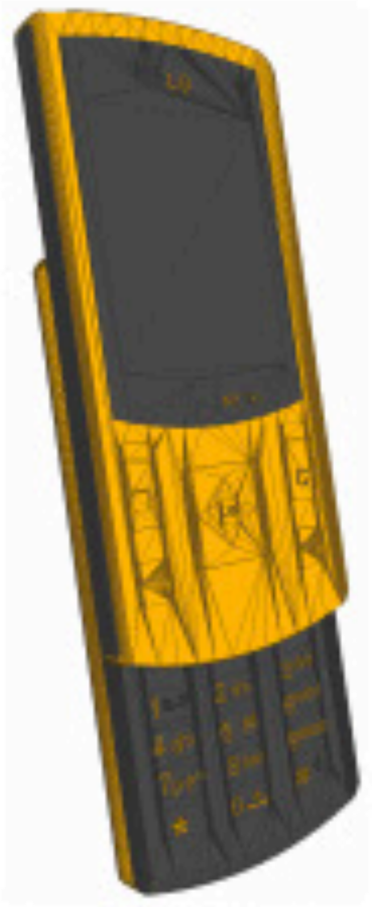


Source: Sacht et. al 2013

non-manifold **not watertight** **intersections**

High bar for geometric computation

non-manifold



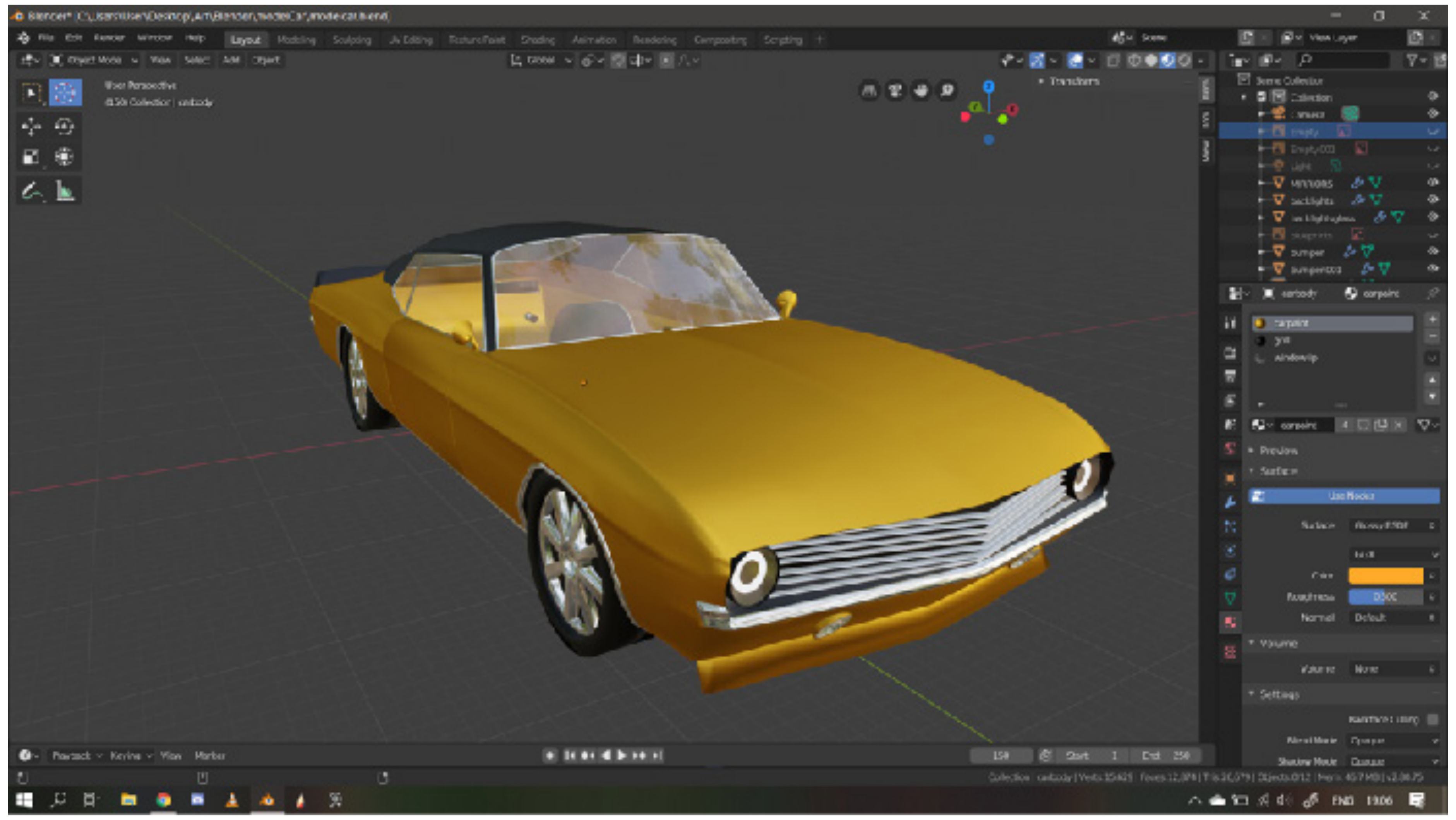
not watertight

intersections

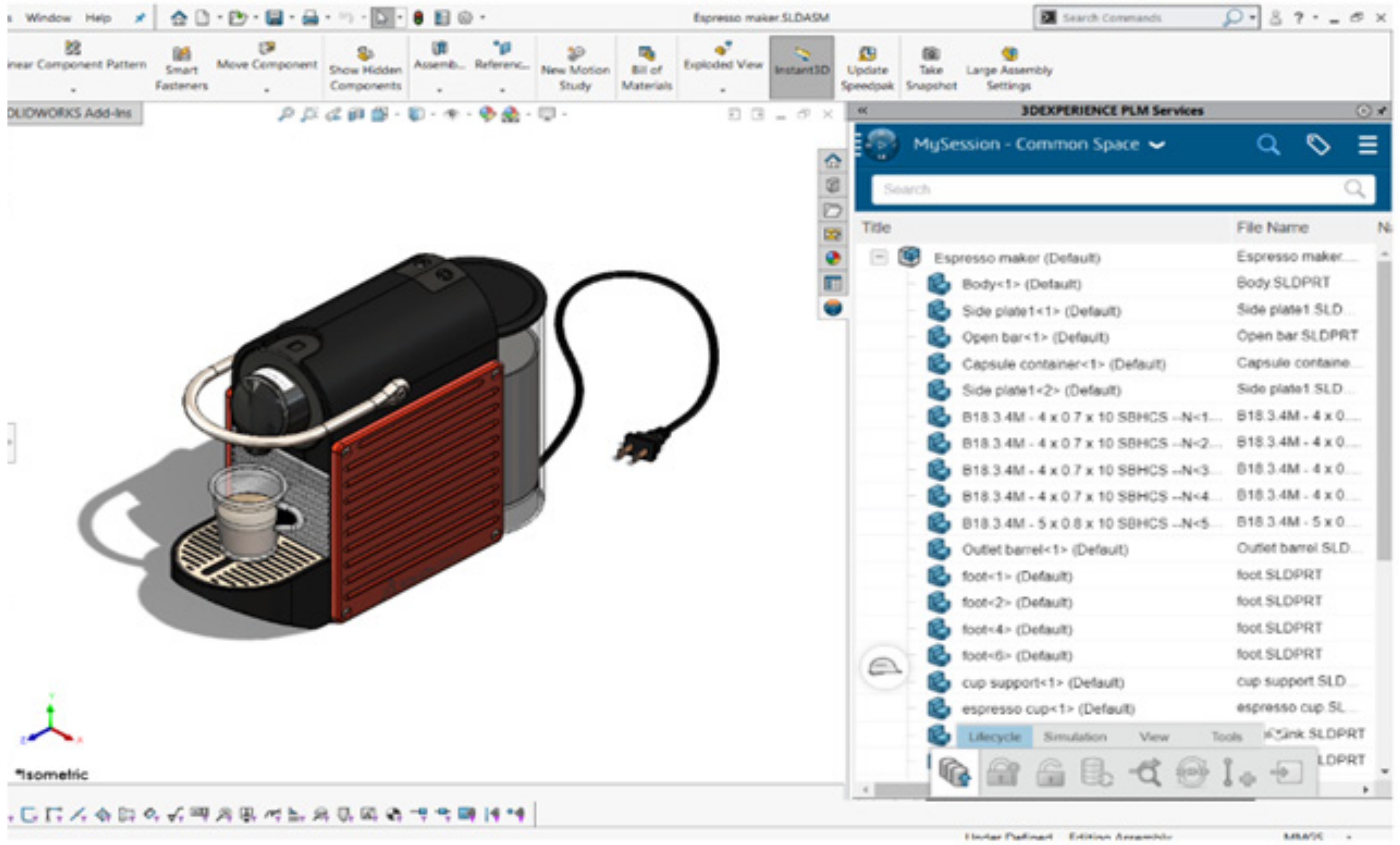
face orientation

Source: Takayama et. al 2014

Hard to Create 3D Data



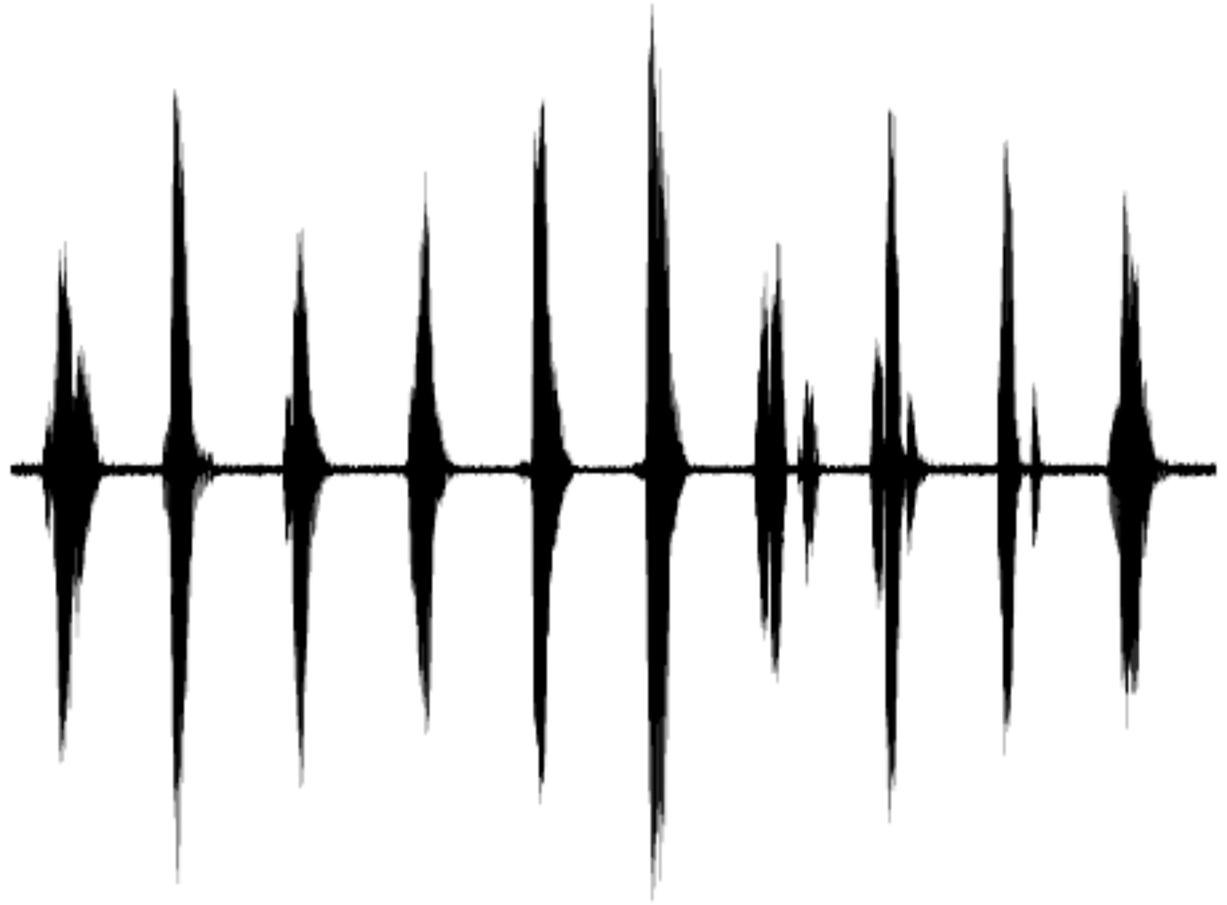
Blender

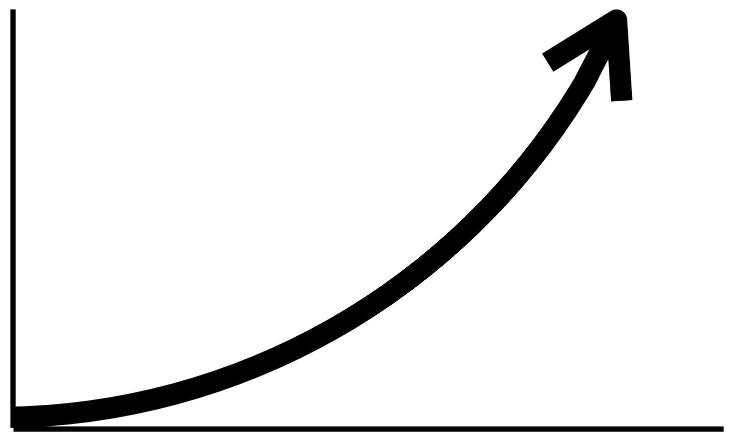


Title	File Name
Espresso maker (Default)	Espresso maker...
Body<1> (Default)	Body.SLDPRT
Side plate1<1> (Default)	Side plate1.SLD...
Open bar<1> (Default)	Open bar.SLDPRT
Capsule container<1> (Default)	Capsule containe...
Side plate1<2> (Default)	Side plate1.SLD...
B18.3.4M - 4 x 0.7 x 10 SBHCS --N<1...	B18.3.4M - 4 x 0...
B18.3.4M - 4 x 0.7 x 10 SBHCS --N<2...	B18.3.4M - 4 x 0...
B18.3.4M - 4 x 0.7 x 10 SBHCS --N<3...	B18.3.4M - 4 x 0...
B18.3.4M - 4 x 0.7 x 10 SBHCS --N<4...	B18.3.4M - 4 x 0...
B18.3.4M - 5 x 0.8 x 10 SBHCS --N<5...	B18.3.4M - 5 x 0...
Outlet barrel<1> (Default)	Outlet barrel.SLD...
foot<1> (Default)	foot.SLDPRT
foot<2> (Default)	foot.SLDPRT
foot<4> (Default)	foot.SLDPRT
foot<6> (Default)	foot.SLDPRT
cup support<1> (Default)	cup support.SLD...
espresso cup<1> (Default)	espresso cup.SL...

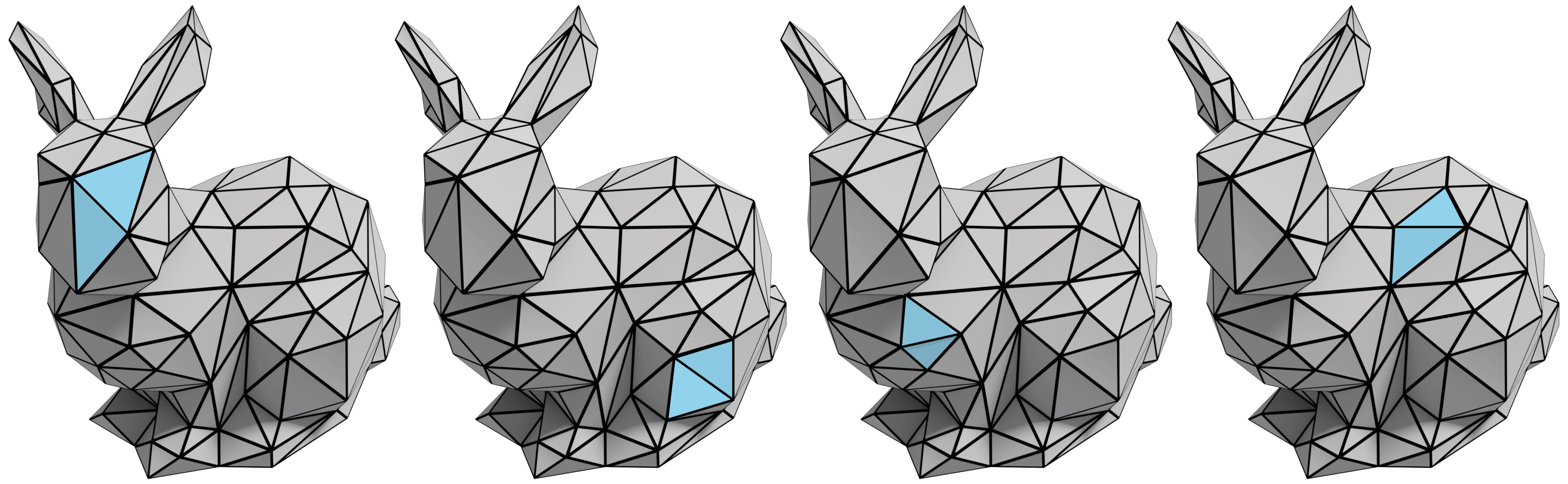
SolidWorks

Curse of dimensionality



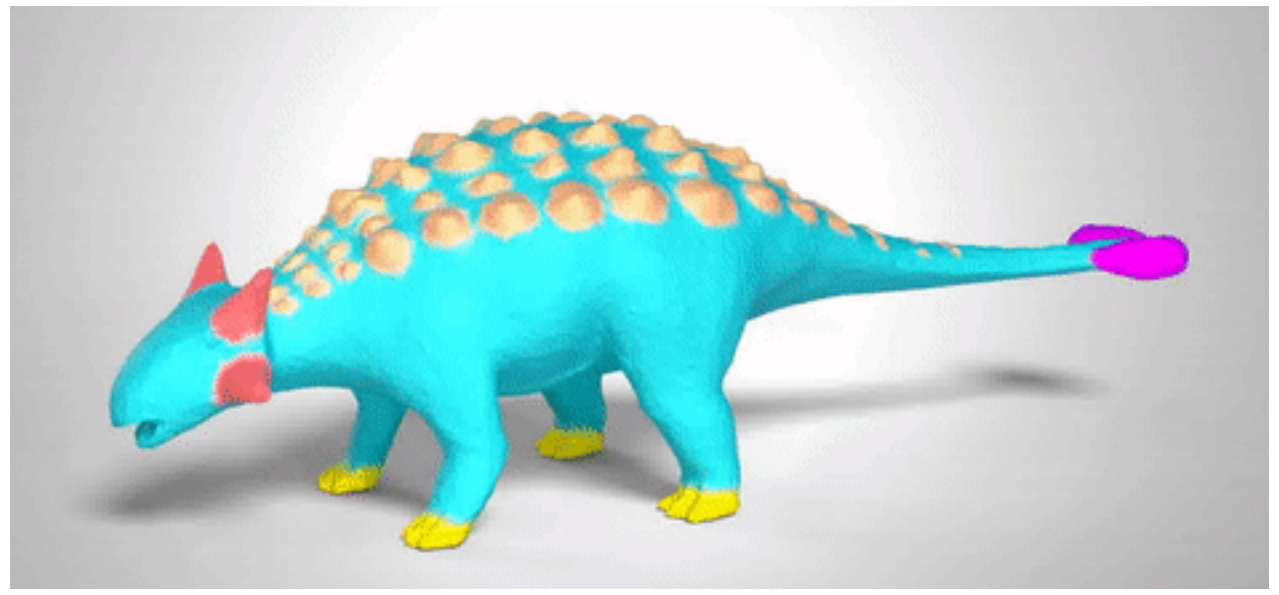


Mesh Convolutional Neural Networks

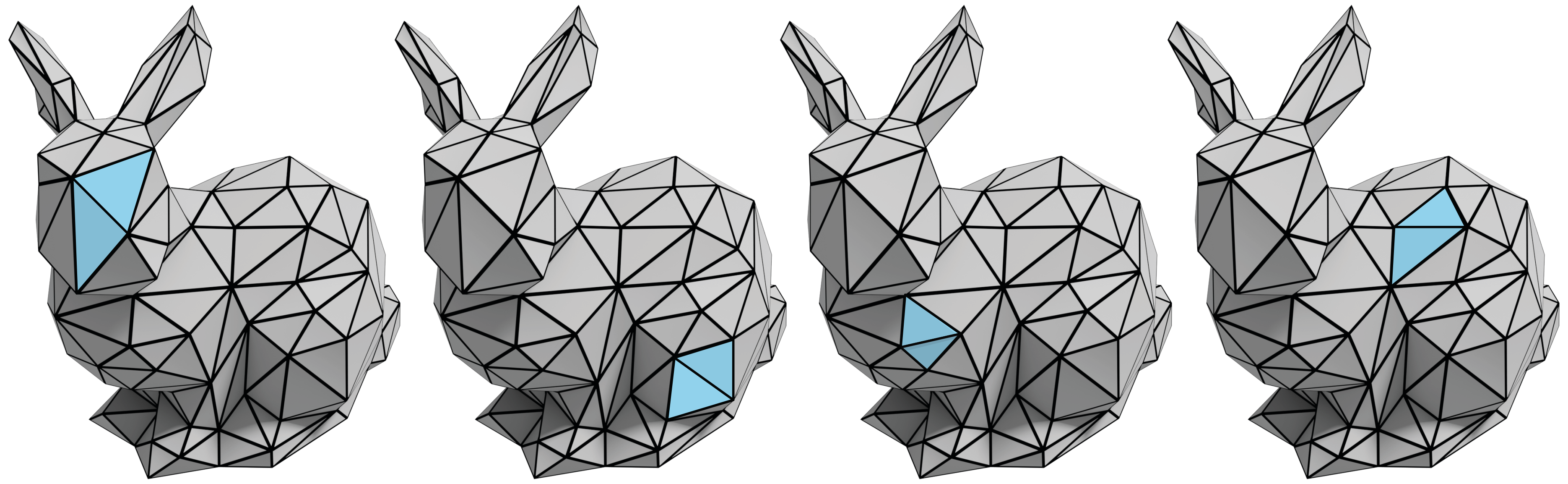


Machine Learning & Geometry Processing

Learning from a Single Mesh

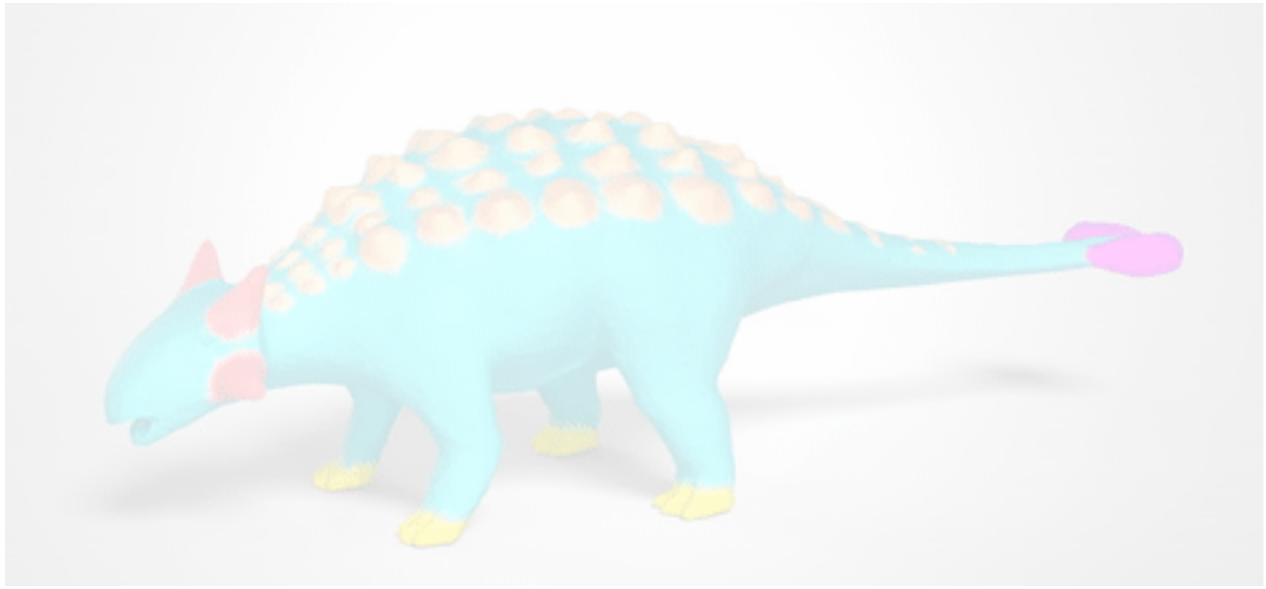


Mesh Convolutional Neural Networks

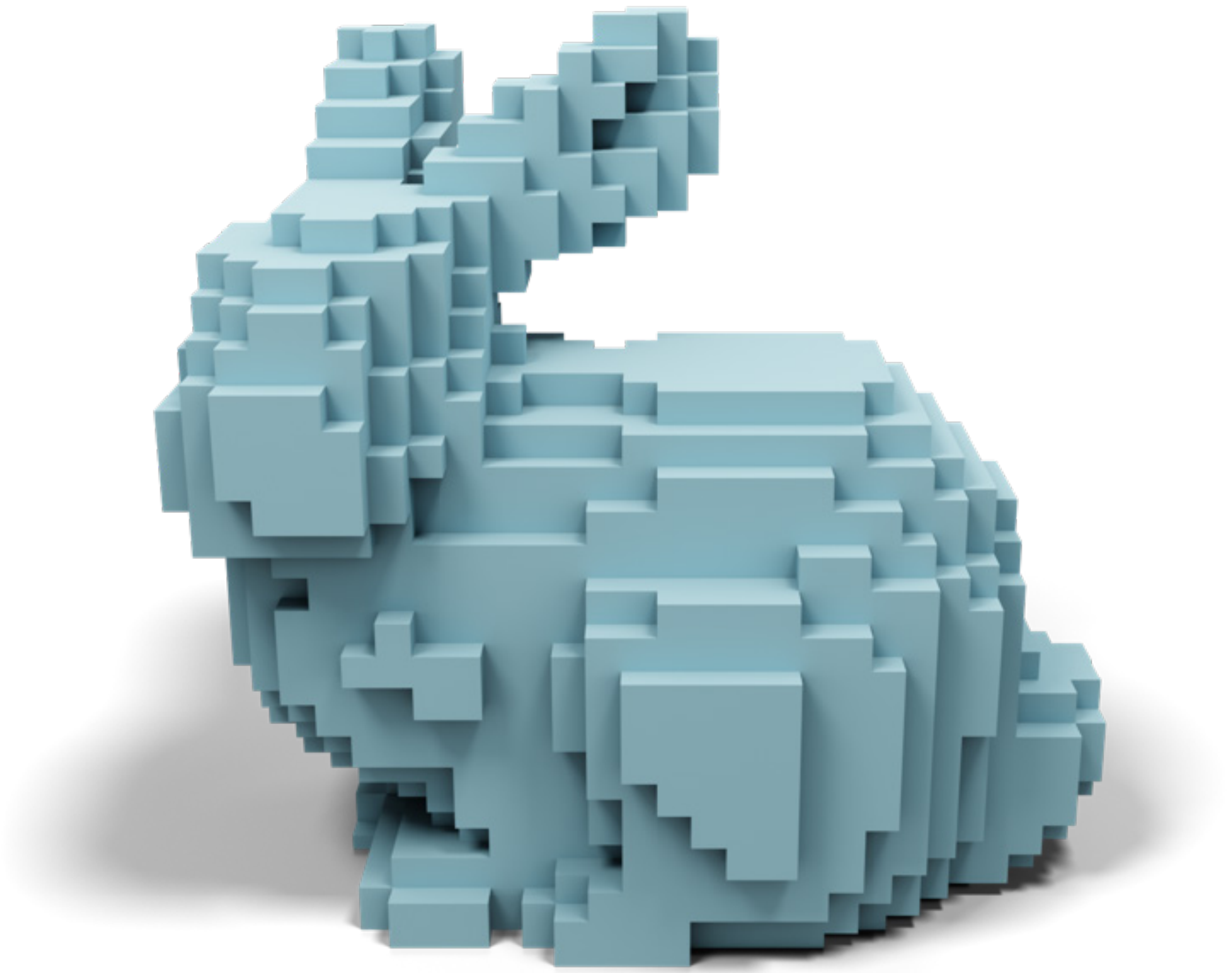


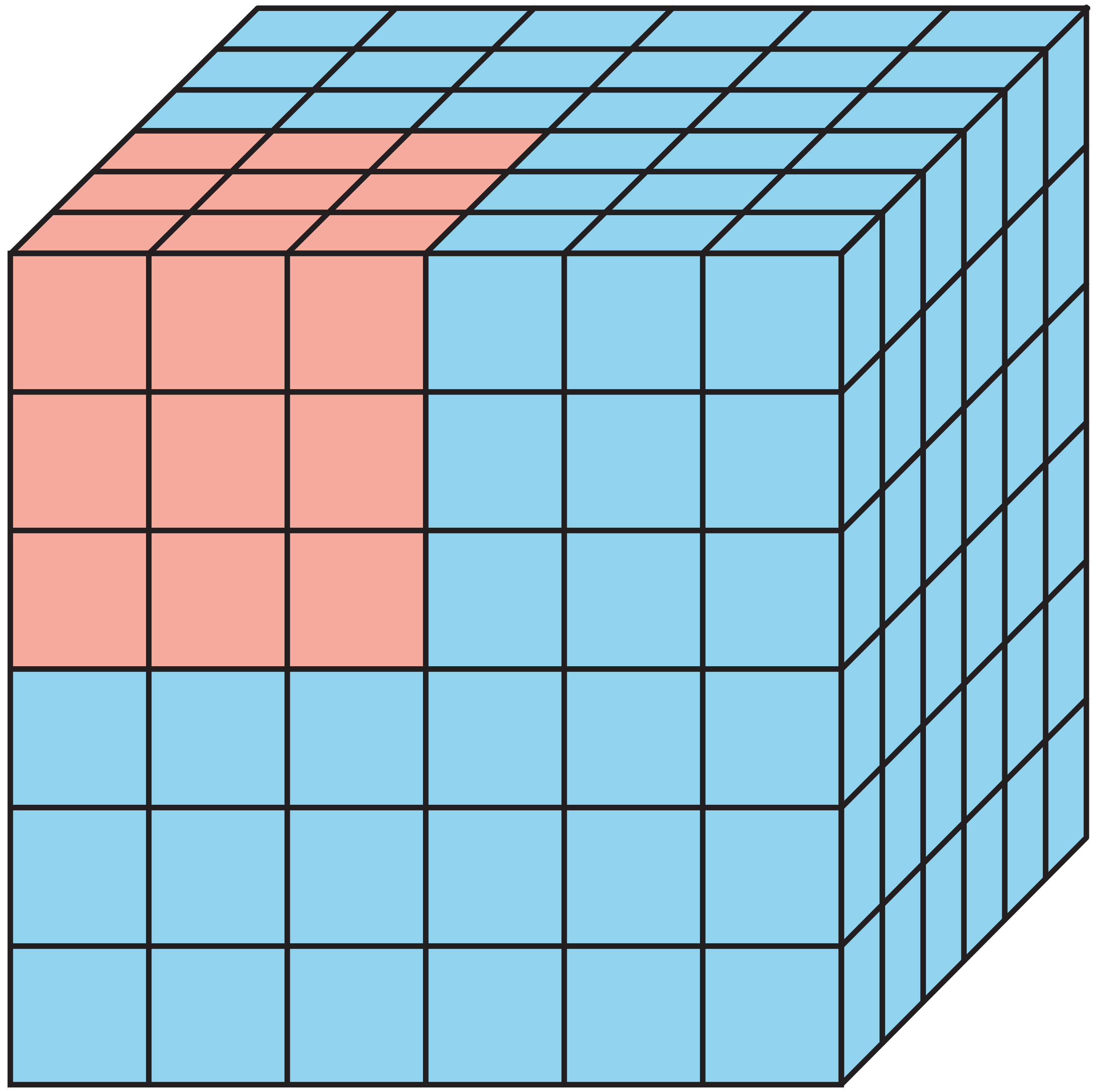
Machine Learning & Geometry Processing

Learning from a Single Mesh



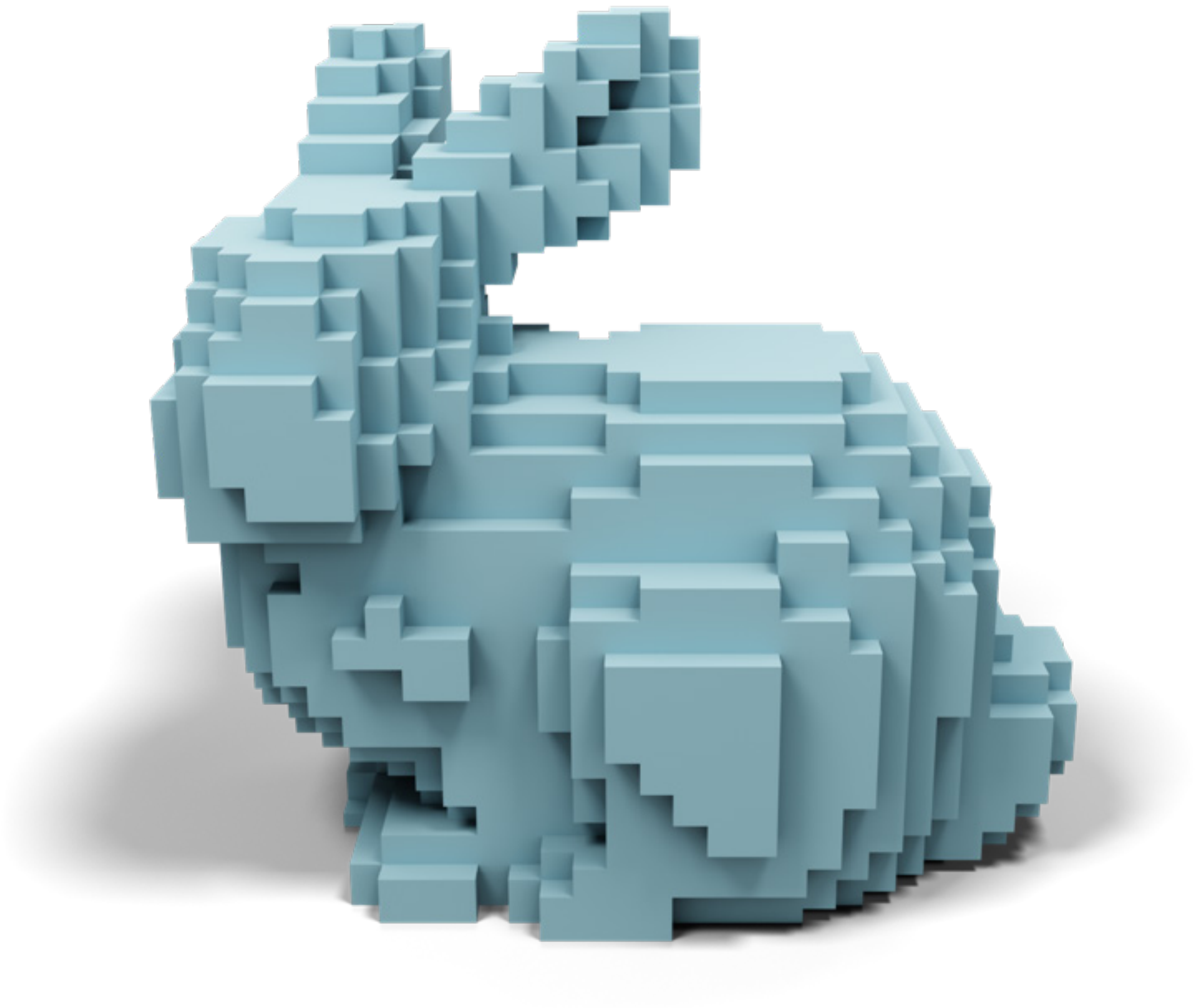
Voxel (3D Pixel) Representation

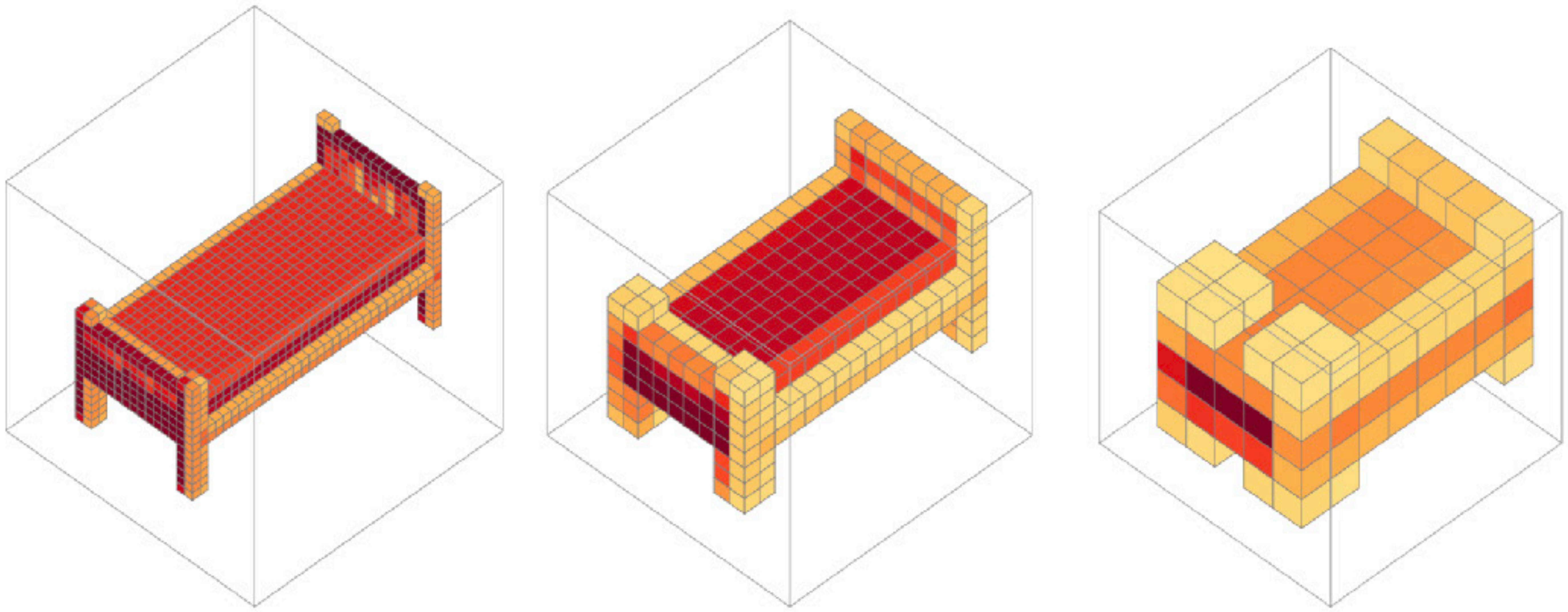




convolution

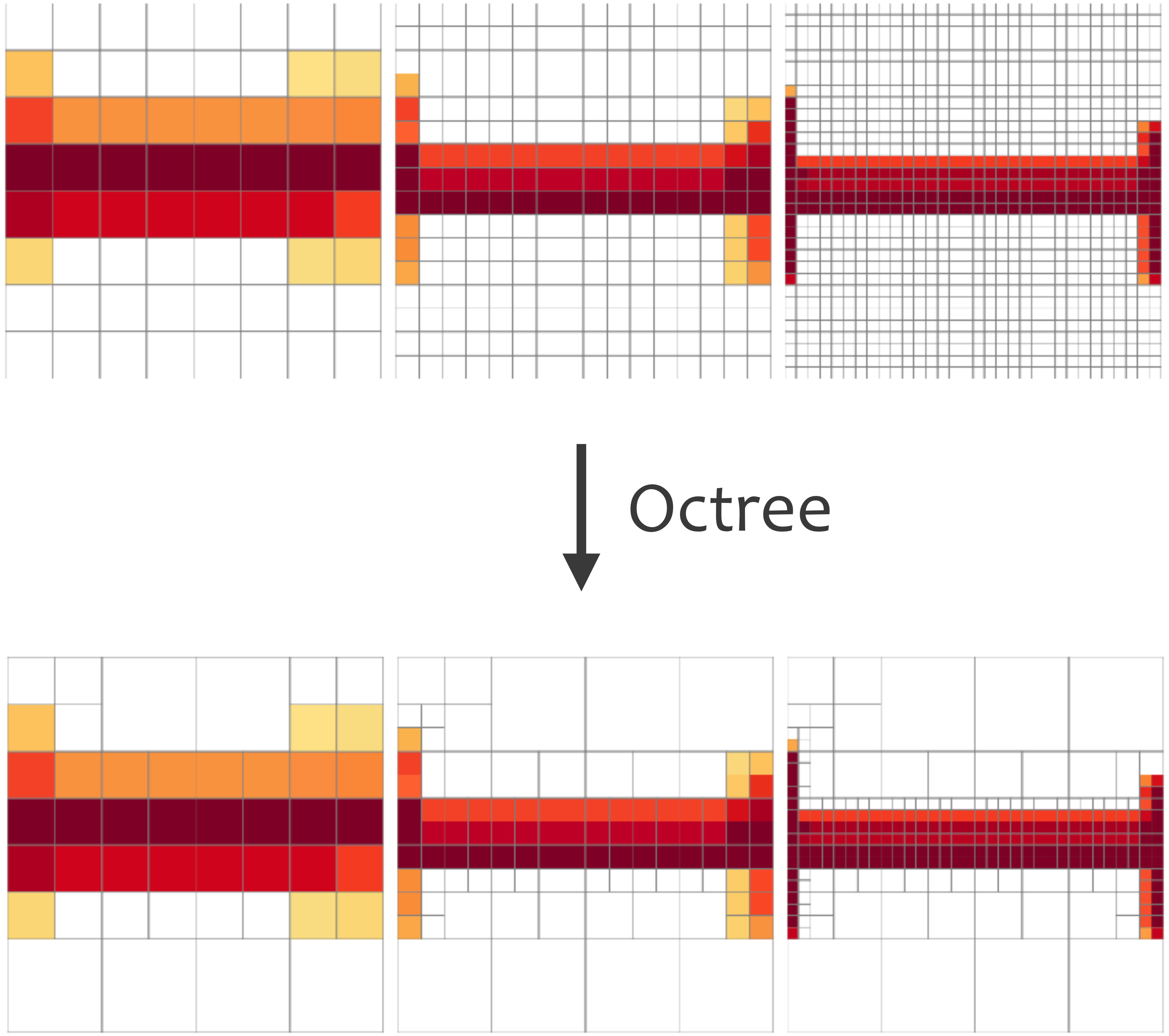
Voxel (3D Pixel) Representation

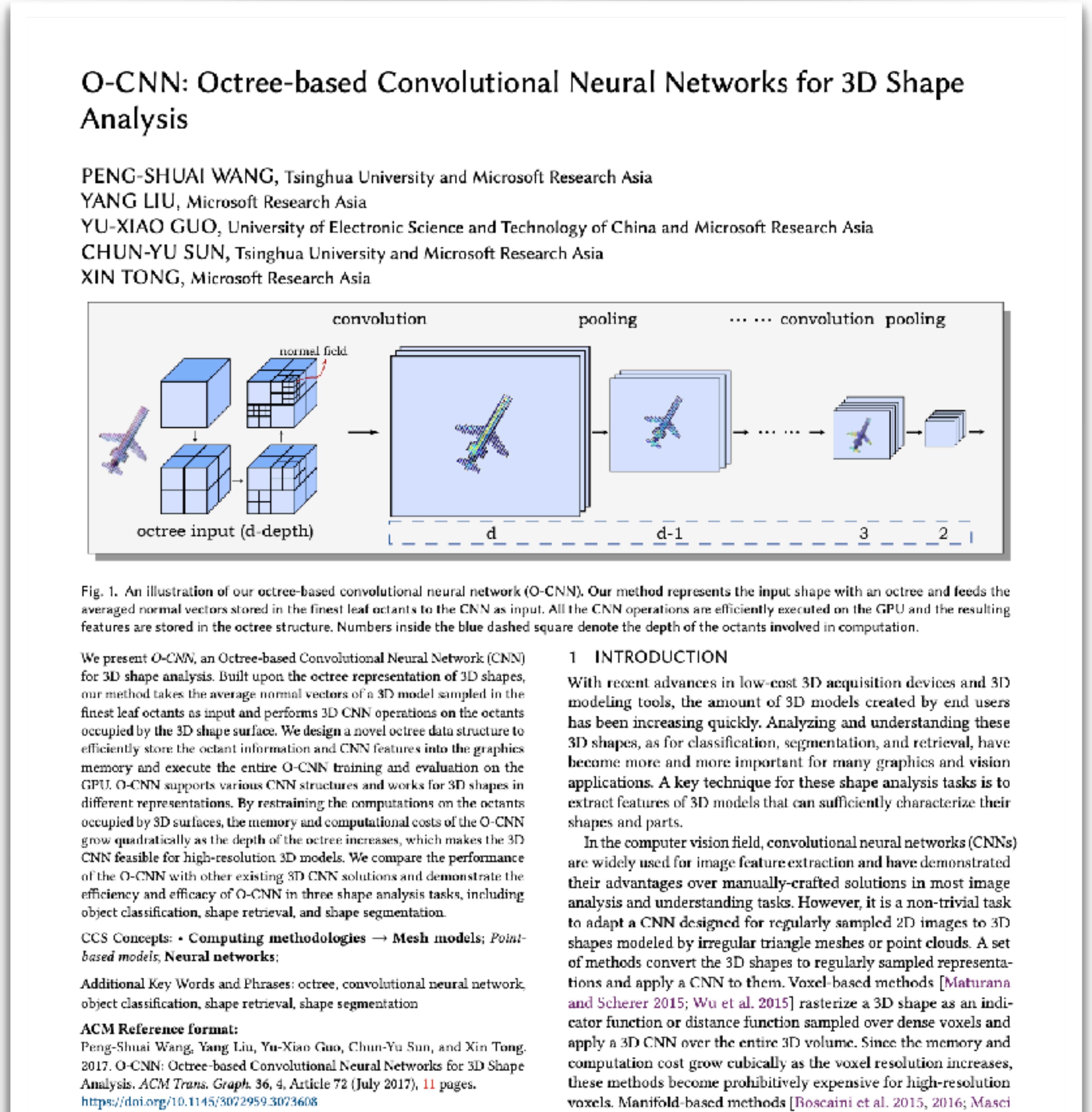




pooling

Large Memory Cost





O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis

PENG-SHUAI WANG, Tsinghua University and Microsoft Research Asia
YANG LIU, Microsoft Research Asia
YU-XIAO GUO, University of Electronic Science and Technology of China and Microsoft Research Asia
CHUN-YU SUN, Tsinghua University and Microsoft Research Asia
XIN TONG, Microsoft Research Asia

Fig. 1. An illustration of our octree-based convolutional neural network (O-CNN). Our method represents the input shape with an octree and feeds the averaged normal vectors stored in the finest leaf octants to the CNN as input. All the CNN operations are efficiently executed on the GPU and the resulting features are stored in the octree structure. Numbers inside the blue dashed square denote the depth of the octants involved in computation.

We present *O-CNN*, an Octree-based Convolutional Neural Network (CNN) for 3D shape analysis. Built upon the octree representation of 3D shapes, our method takes the average normal vectors of a 3D model sampled in the finest leaf octants as input and performs 3D CNN operations on the octants occupied by the 3D shape surface. We design a novel octree data structure to efficiently store the octant information and CNN features into the graphics memory and execute the entire O-CNN training and evaluation on the GPU. O-CNN supports various CNN structures and works for 3D shapes in different representations. By restraining the computations on the octants occupied by 3D surfaces, the memory and computational costs of the O-CNN grow quadratically as the depth of the octree increases, which makes the 3D CNN feasible for high-resolution 3D models. We compare the performance of the O-CNN with other existing 3D CNN solutions and demonstrate the efficiency and efficacy of O-CNN in three shape analysis tasks, including object classification, shape retrieval, and shape segmentation.

CCS Concepts: • **Computing methodologies** → **Mesh models**; *Point-based models*; **Neural networks**;

Additional Key Words and Phrases: octree, convolutional neural network, object classification, shape retrieval, shape segmentation

ACM Reference format:
Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.* 36, 4, Article 72 (July 2017), 11 pages.
https://doi.org/10.1145/3072959.3073608

1 INTRODUCTION

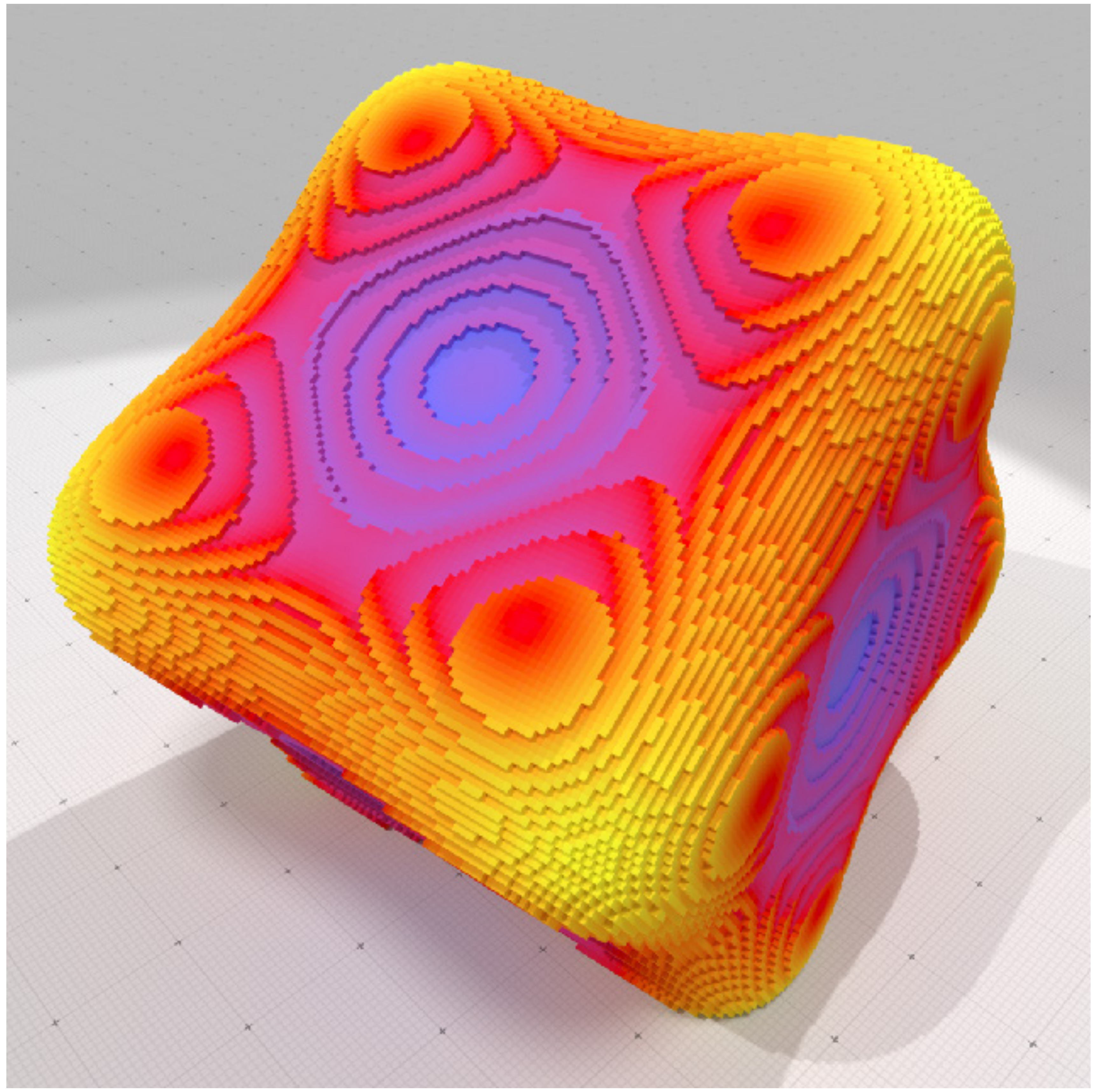
With recent advances in low-cost 3D acquisition devices and 3D modeling tools, the amount of 3D models created by end users has been increasing quickly. Analyzing and understanding these 3D shapes, as for classification, segmentation, and retrieval, have become more and more important for many graphics and vision applications. A key technique for these shape analysis tasks is to extract features of 3D models that can sufficiently characterize their shapes and parts.

In the computer vision field, convolutional neural networks (CNNs) are widely used for image feature extraction and have demonstrated their advantages over manually-crafted solutions in most image analysis and understanding tasks. However, it is a non-trivial task to adapt a CNN designed for regularly sampled 2D images to 3D shapes modeled by irregular triangle meshes or point clouds. A set of methods convert the 3D shapes to regularly sampled representations and apply a CNN to them. Voxel-based methods [Maturana and Scherer 2015; Wu et al. 2015] rasterize a 3D shape as an indicator function or distance function sampled over dense voxels and apply a 3D CNN over the entire 3D volume. Since the memory and computation cost grow cubically as the voxel resolution increases, these methods become prohibitively expensive for high-resolution voxels. Manifold-based methods [Boscaini et al. 2015, 2016; Masci

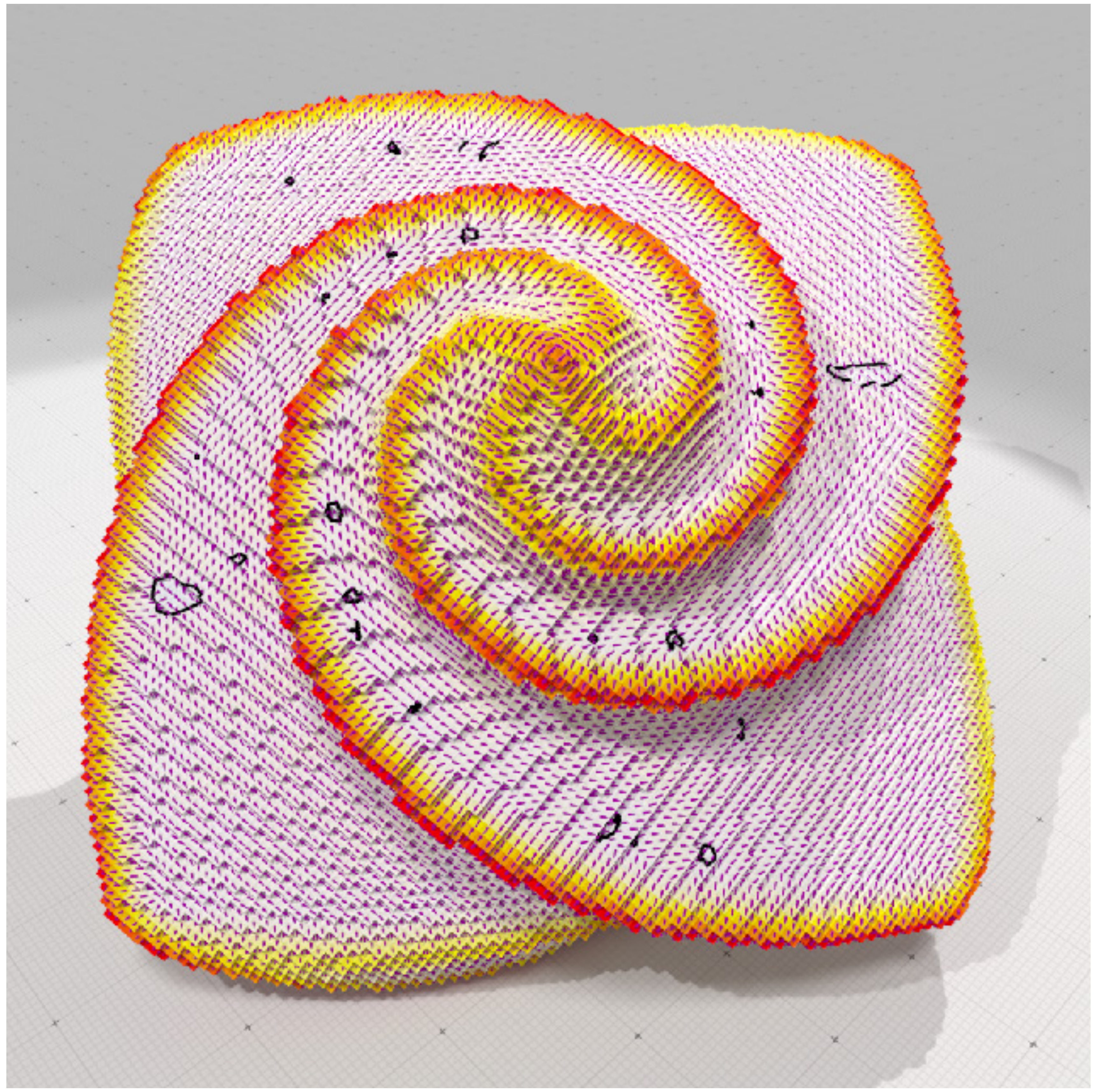
Not a smooth representation

source: minecraft

Estimating surface quantities on voxels

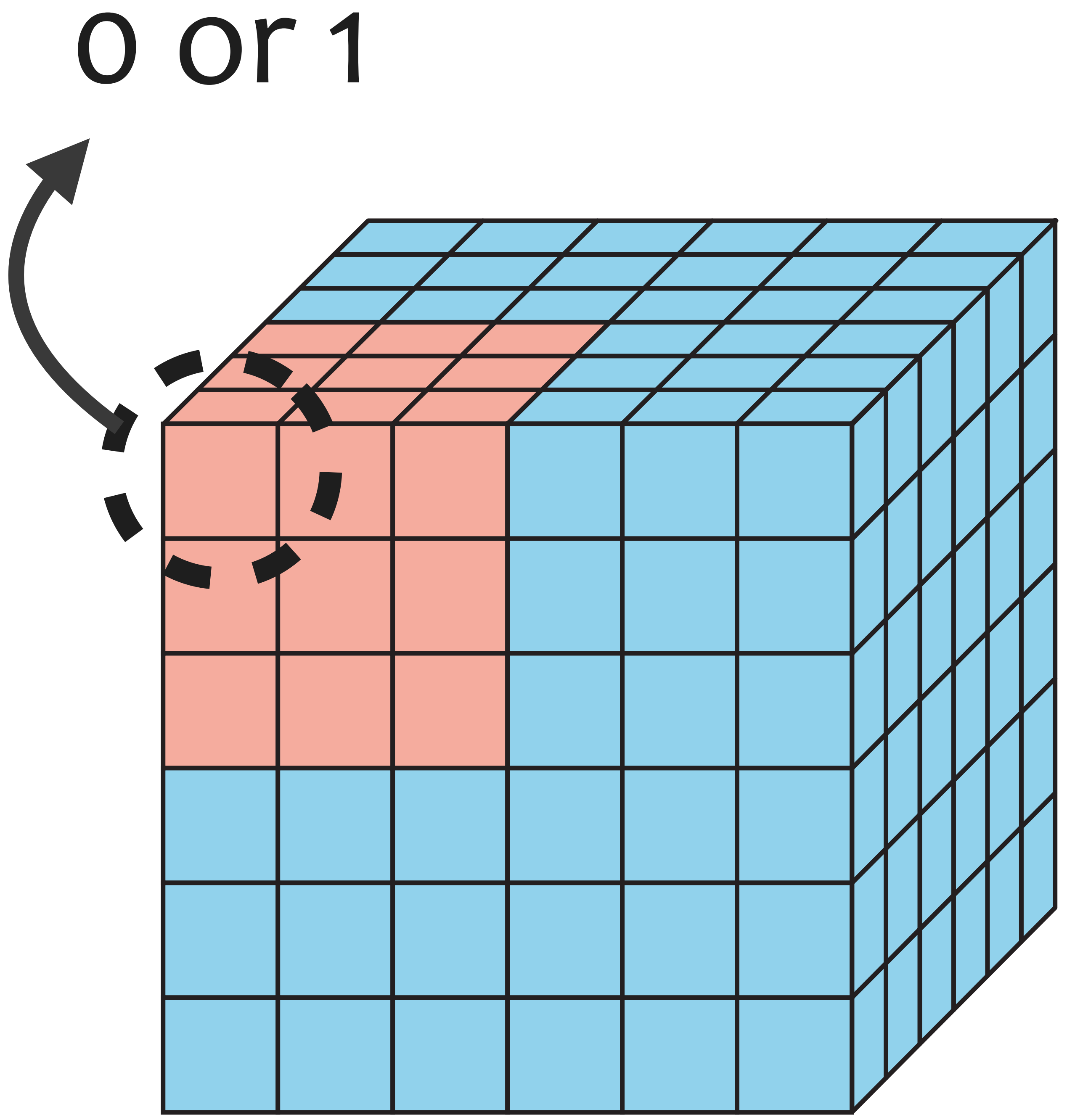


Caissard et al. 2019

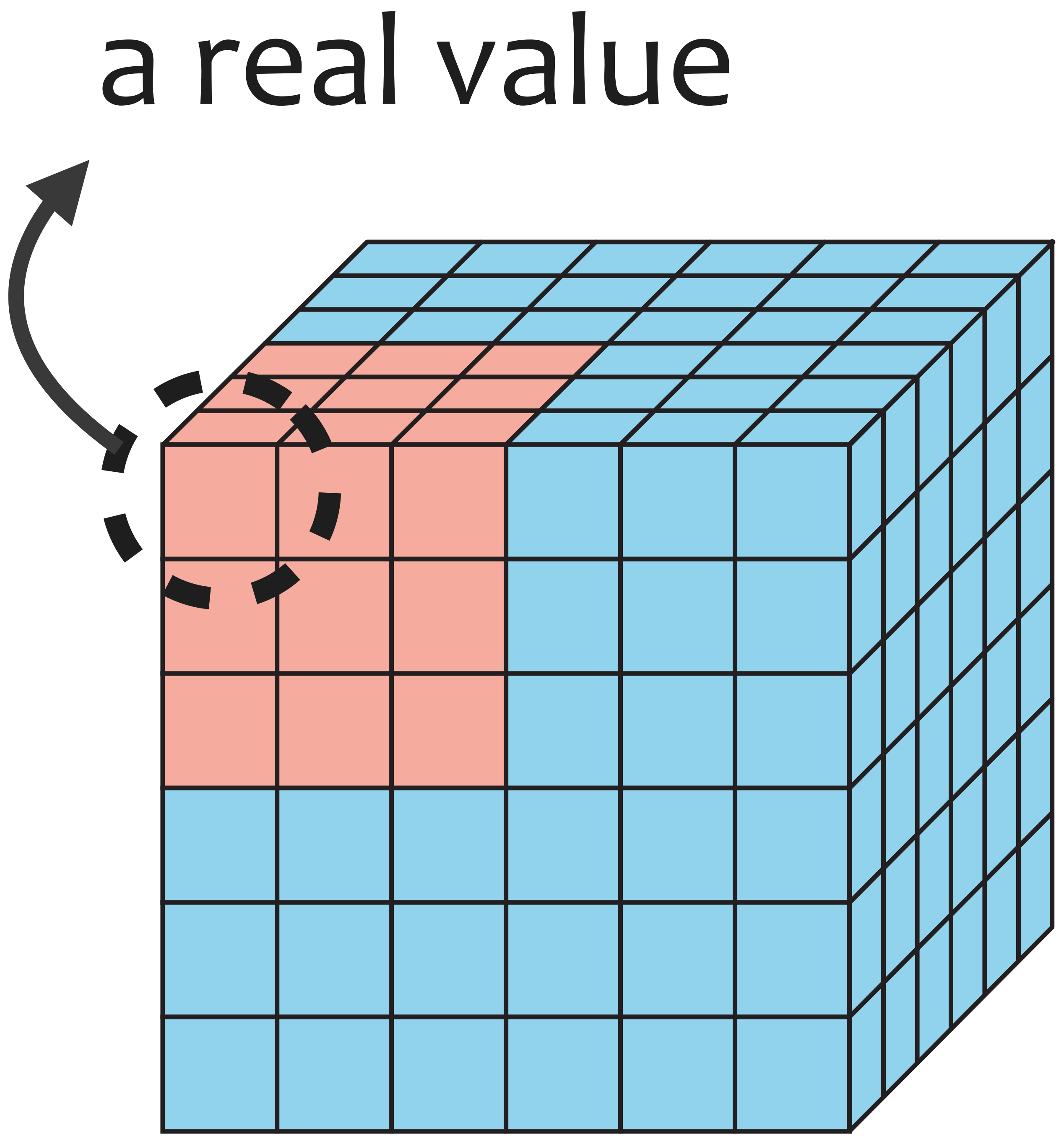


Lachaud et al. 2020

Not store {0,1}



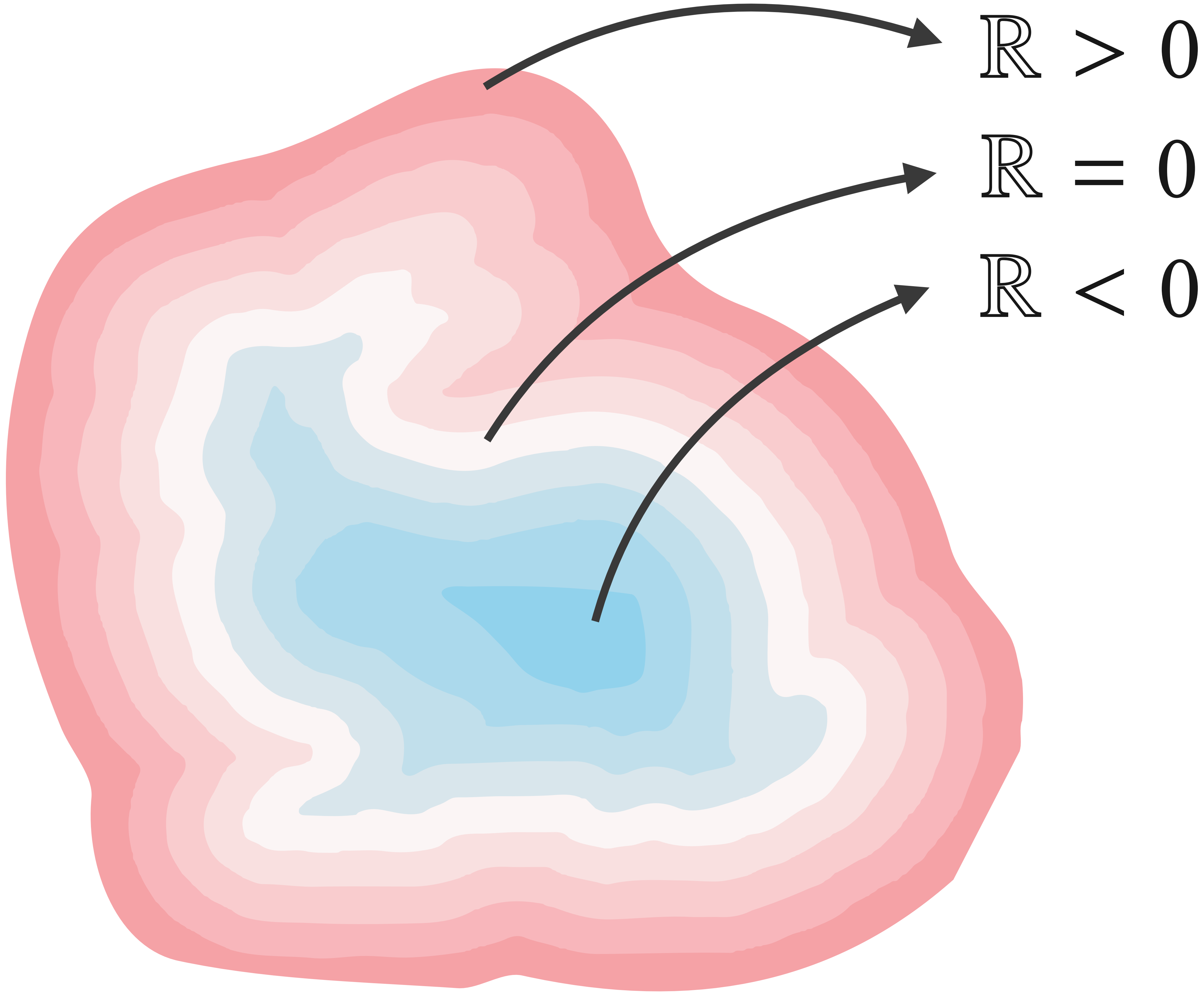
Not store {0,1}



a real value

signed distance function (SDF)

[Dai et al. 2017, Zeng et al. 2017, Stutz et al. 2018]

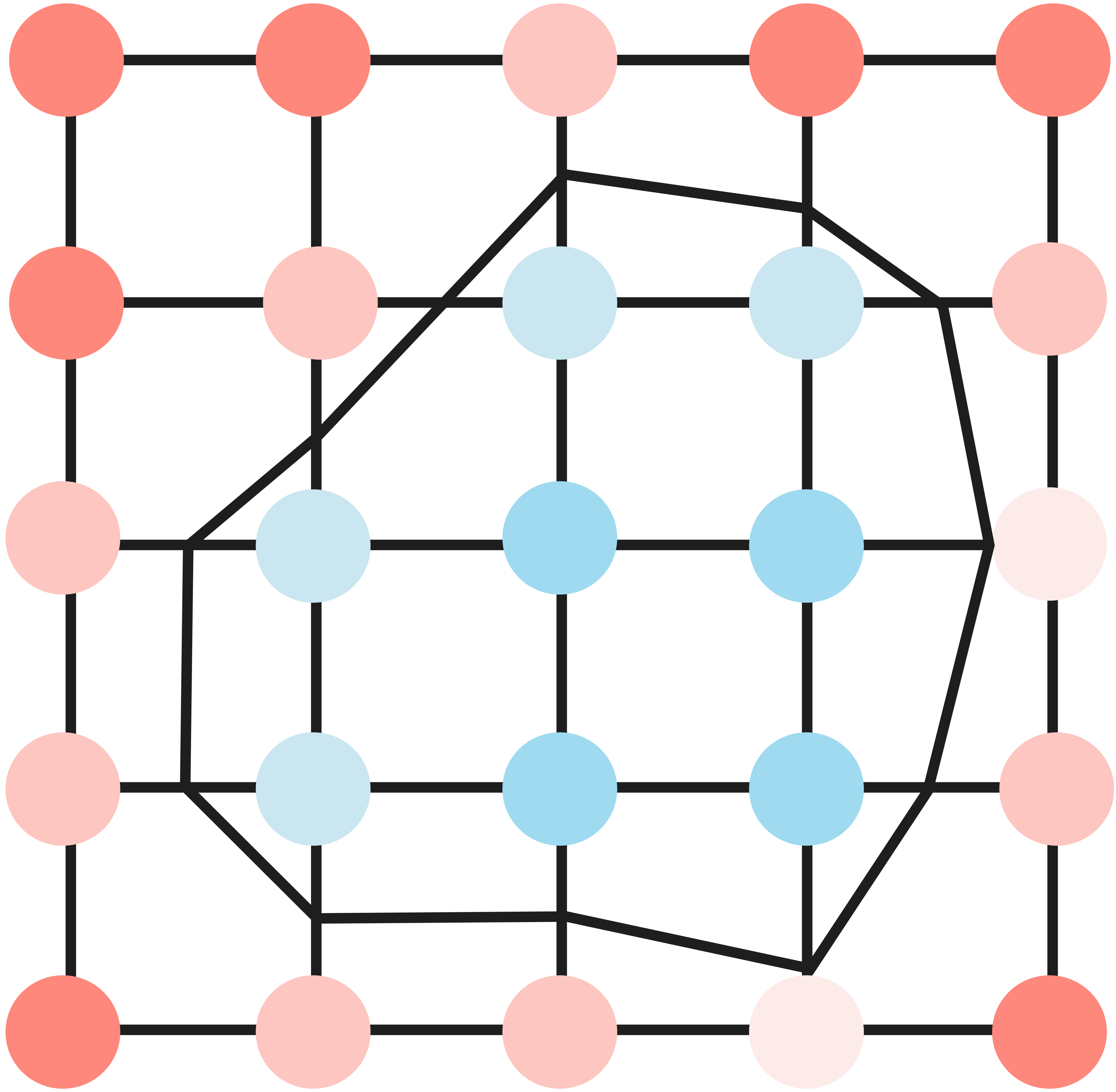


$\mathbb{R} > 0$

$\mathbb{R} = 0$

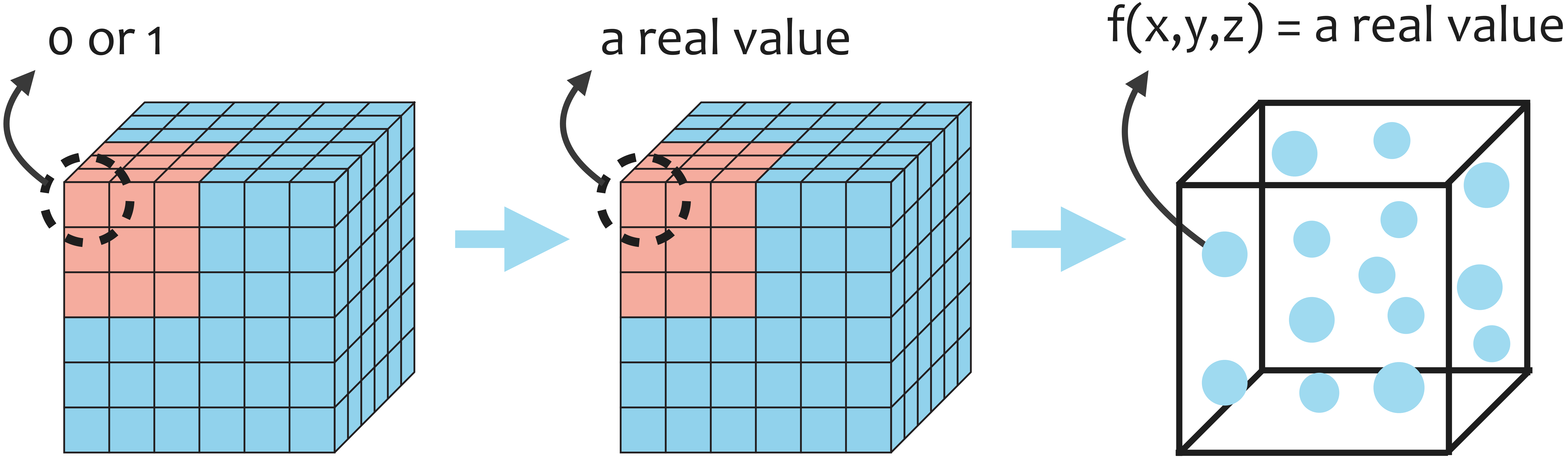
$\mathbb{R} < 0$

SDF to sub-pixel resolution

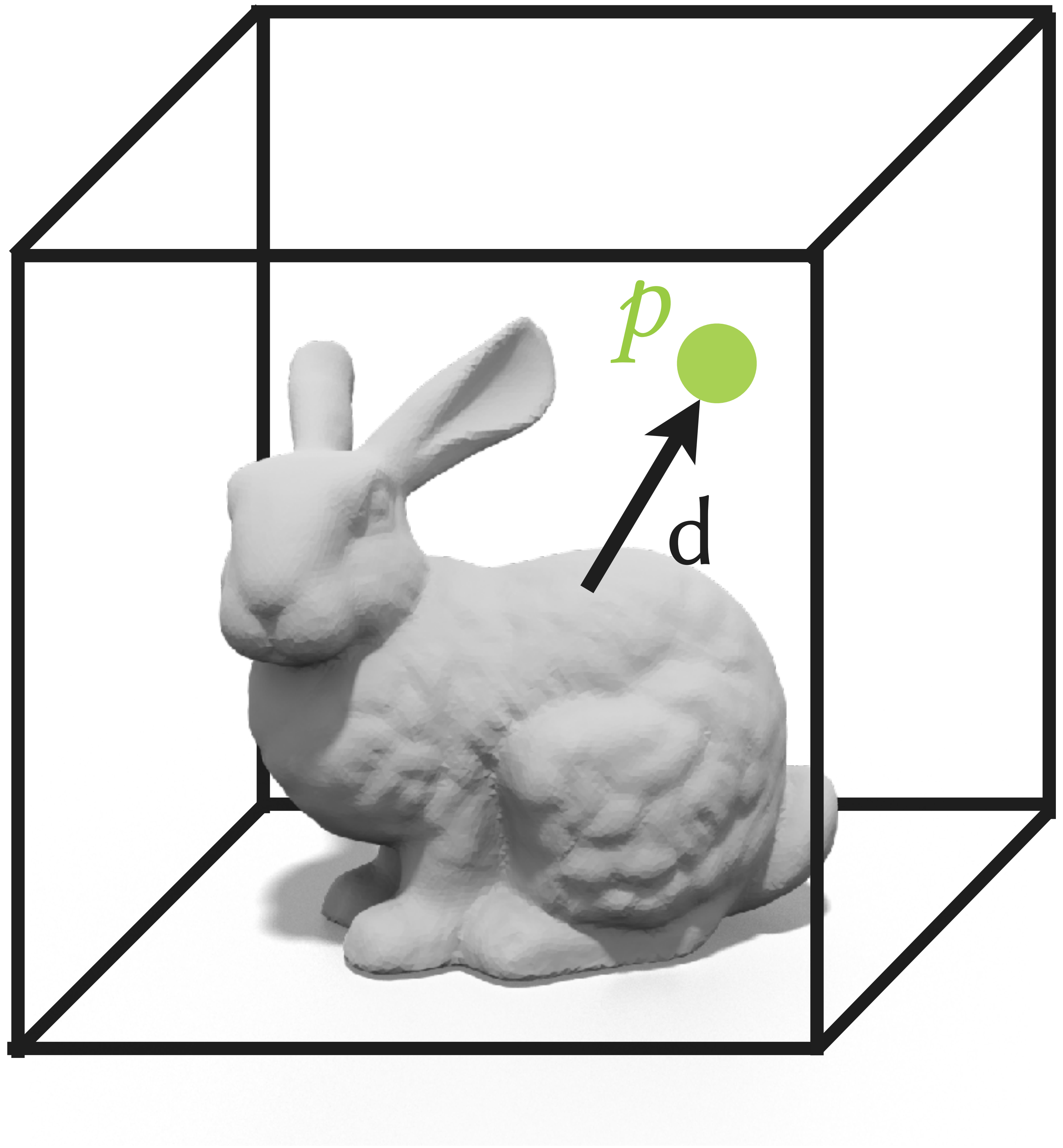


[Dai et al. 2017]

Get rid of the voxel grid

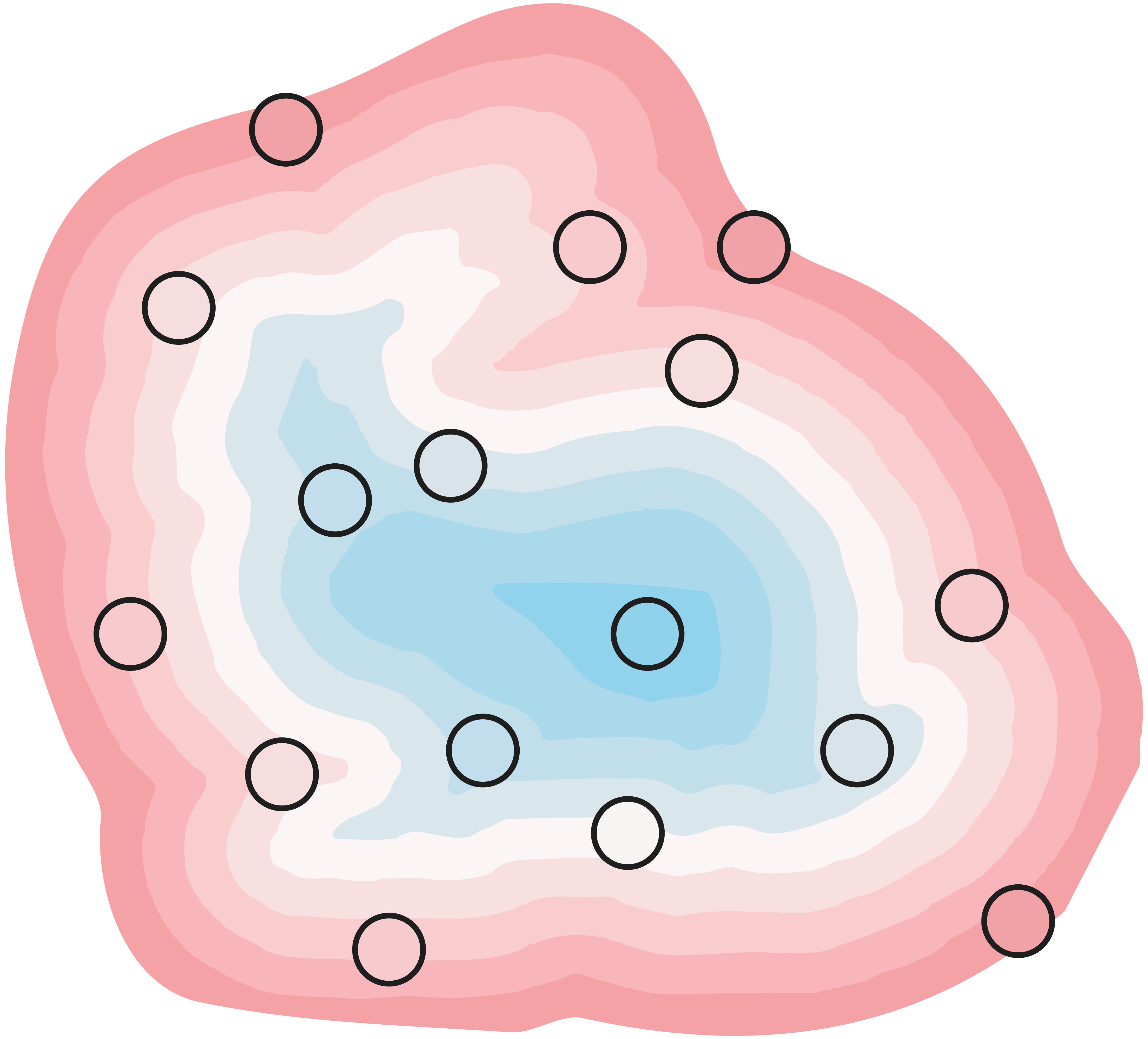


Neural Signed Distance Function



$$f_{\theta}(p_x, p_y, p_z) = \text{signed distance}$$

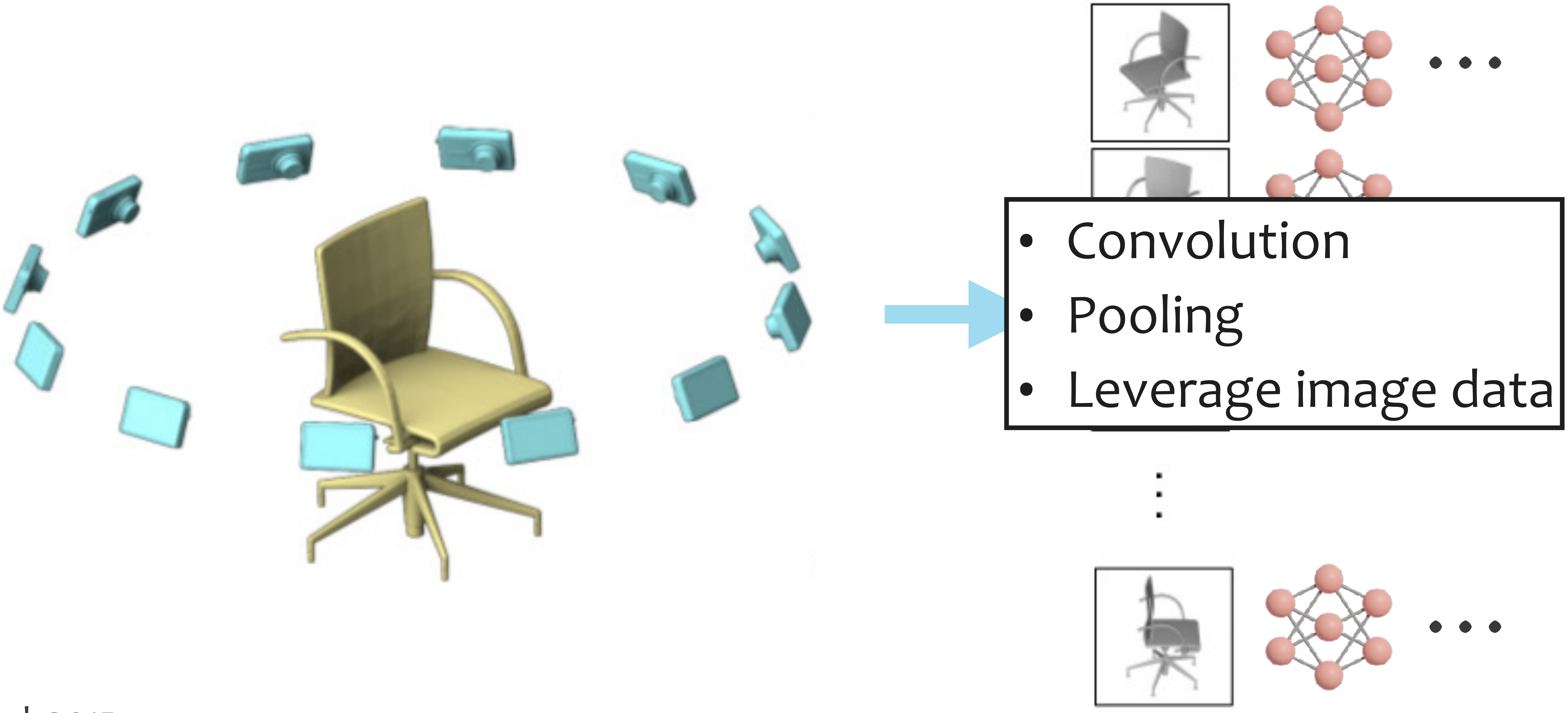
An Alternative Shape Representation



missing basic ingredients:

- differential operators
- differential quantities
- shape editing
- ...

Use image convolution on surfaces

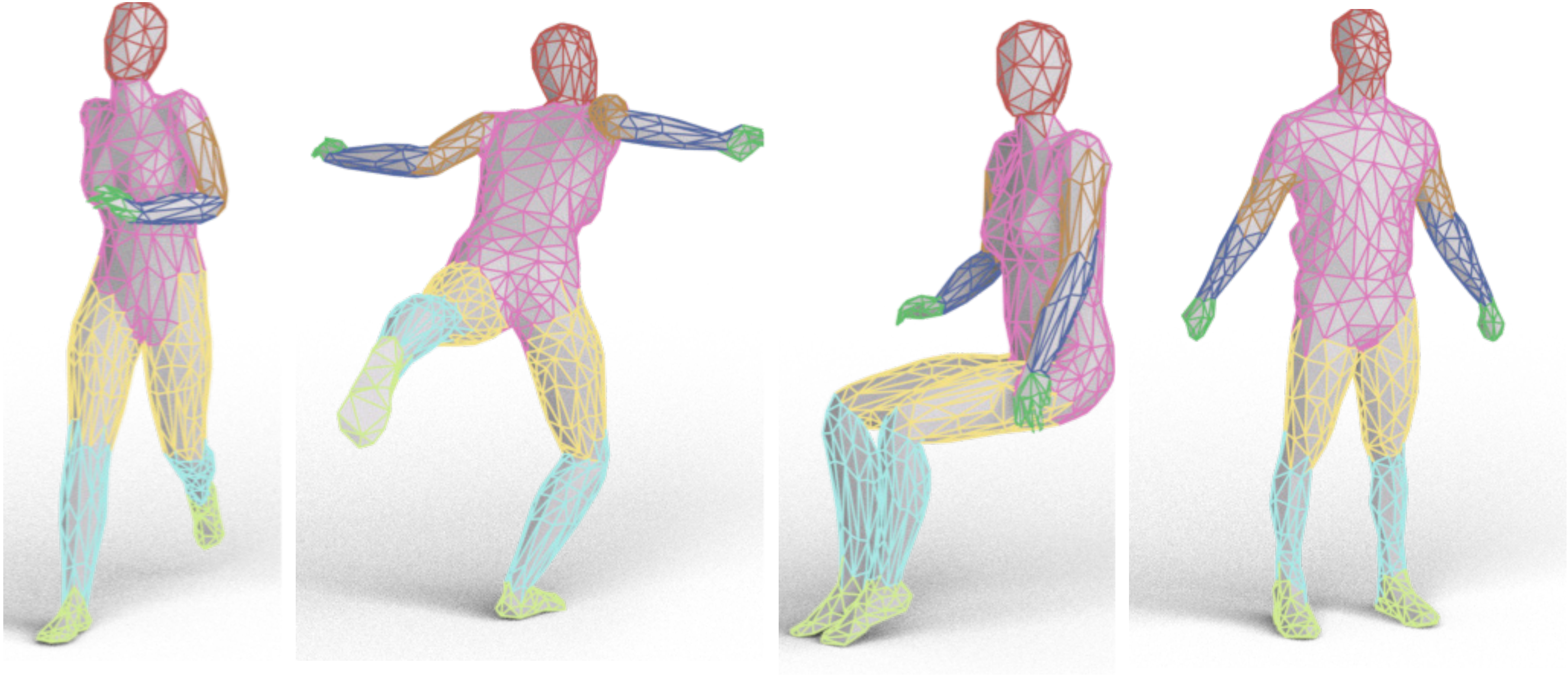


- Convolution
- Pooling
- Leverage image data

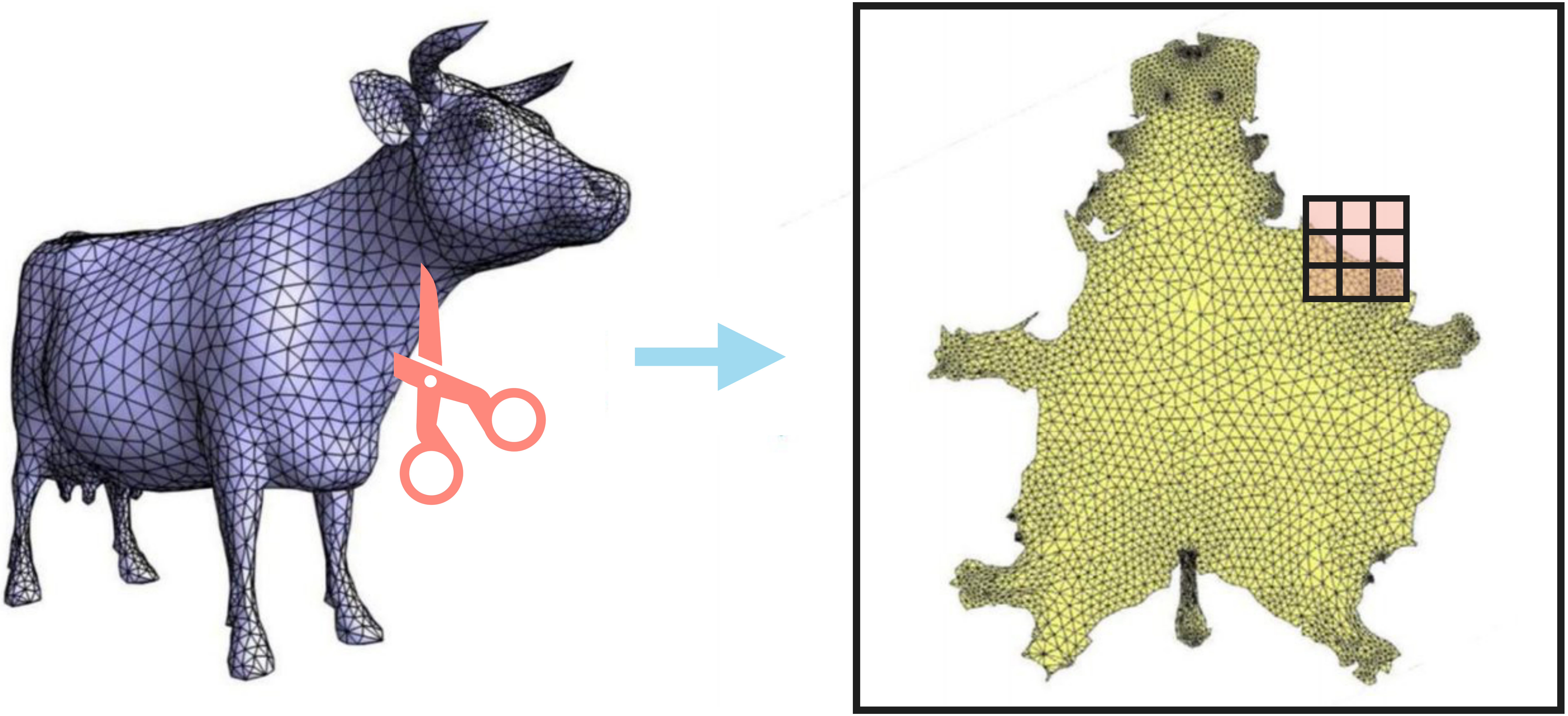
Su et al. 2015

Leverage Image Data

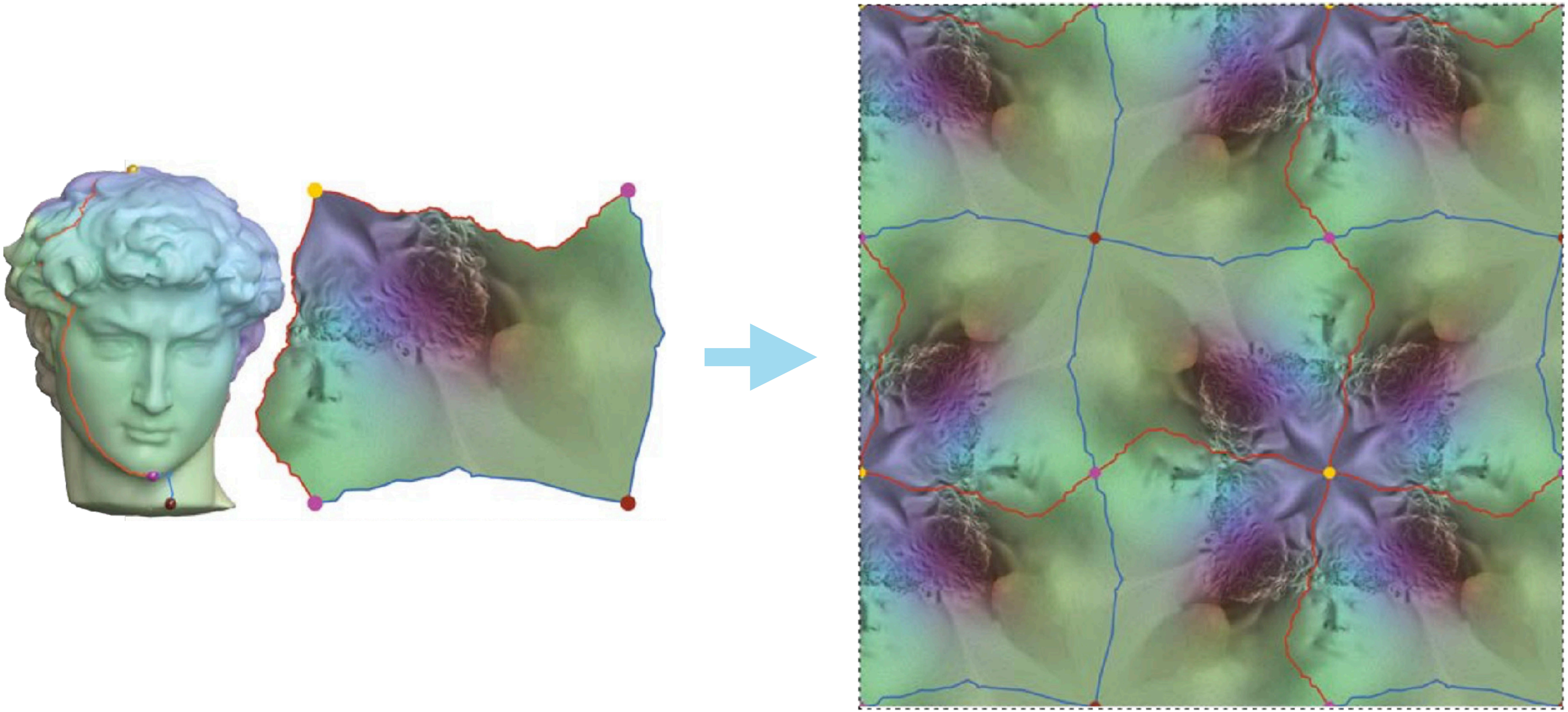
Extending to local tasks is hard



Global Parameterization

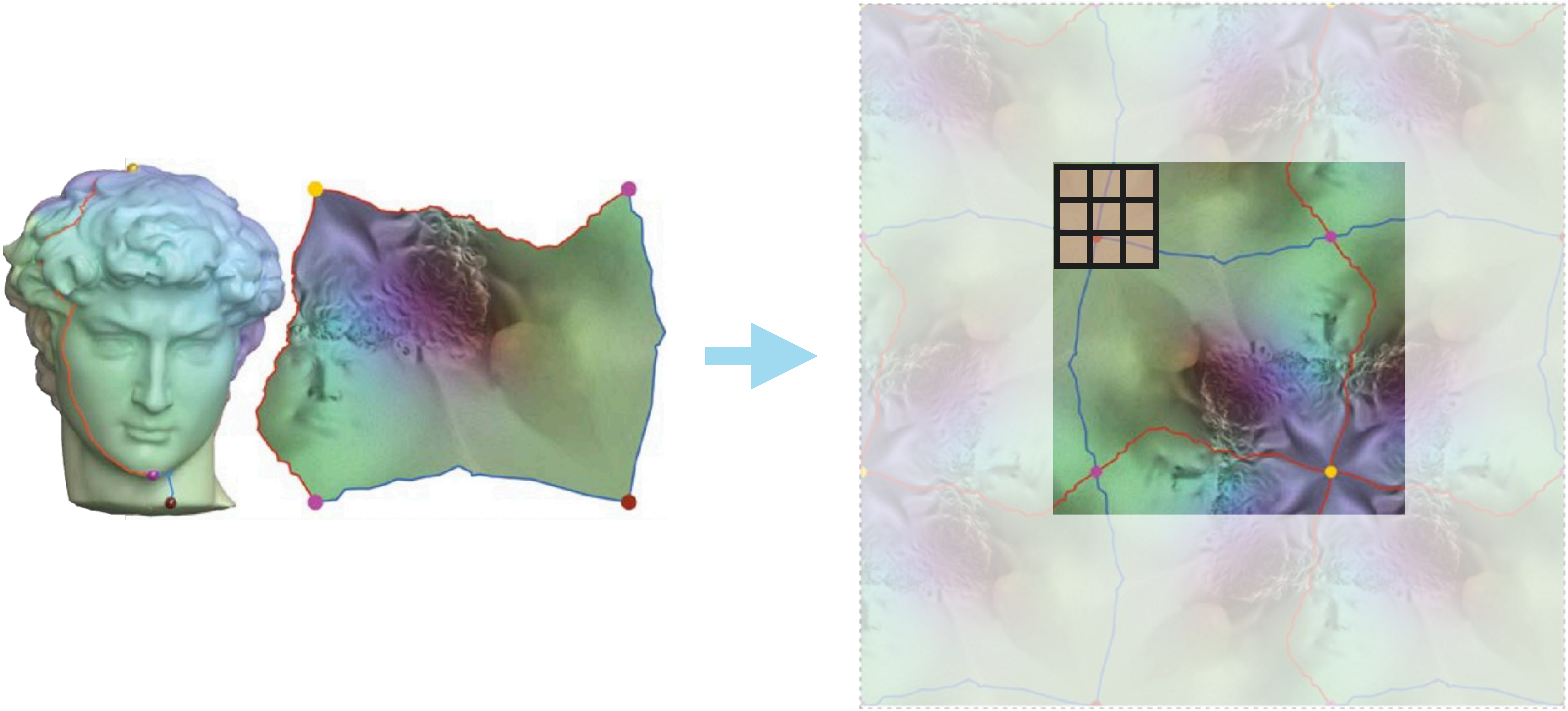


Seamless Parameterization



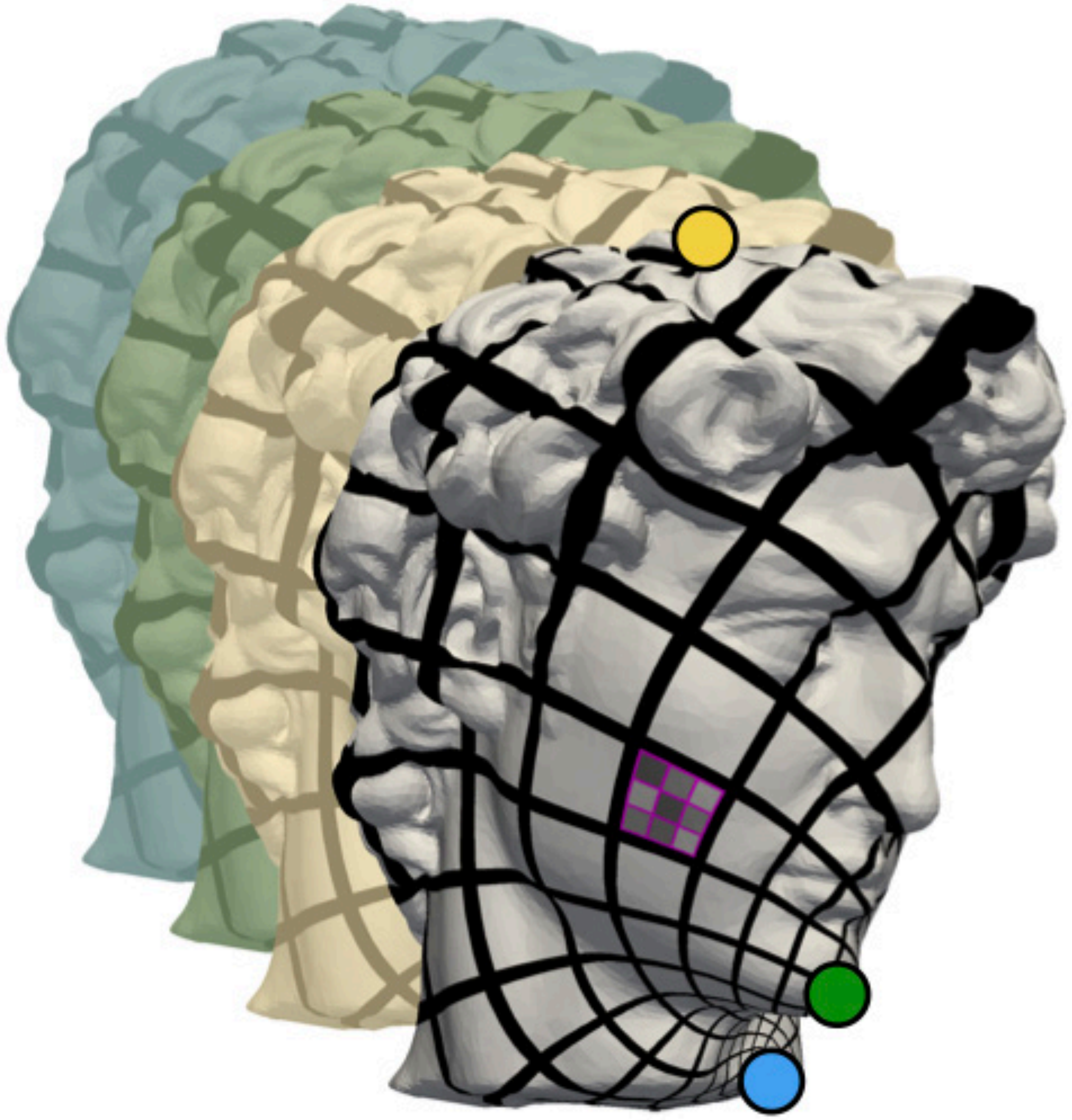
Aigerman & Lipman 2015

Seamless Parameterization

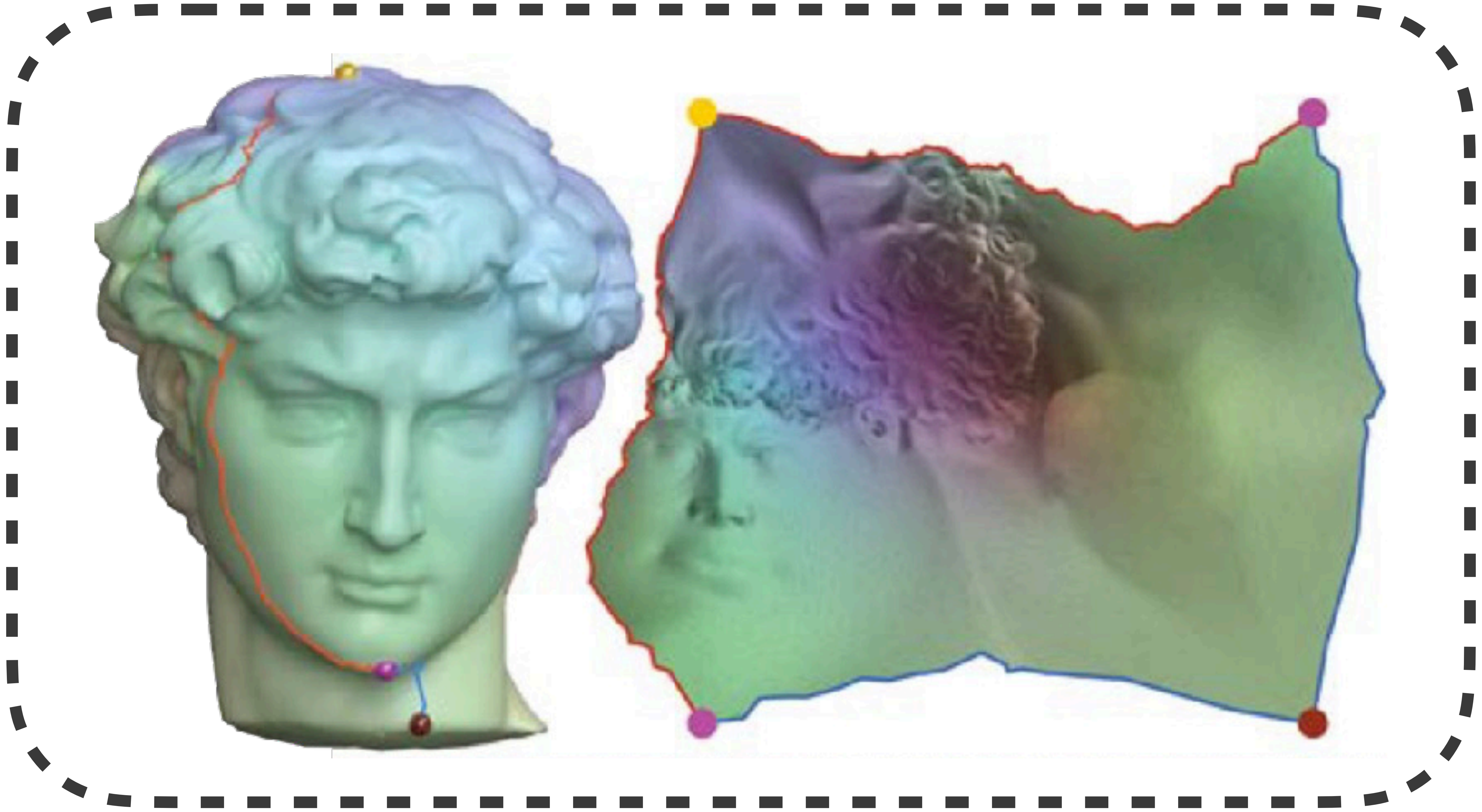


Aigerman & Lipman 2015

Challenges

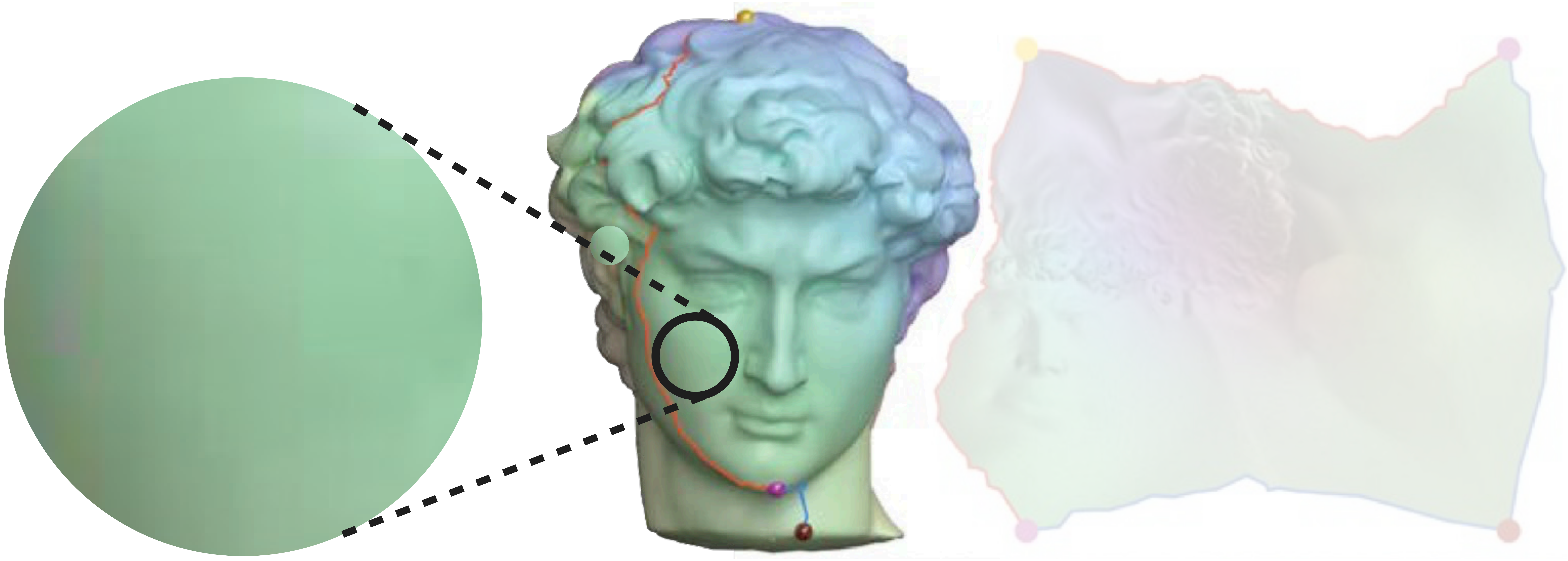


- Not unique
- Orientation
- Cannot avoid distortion

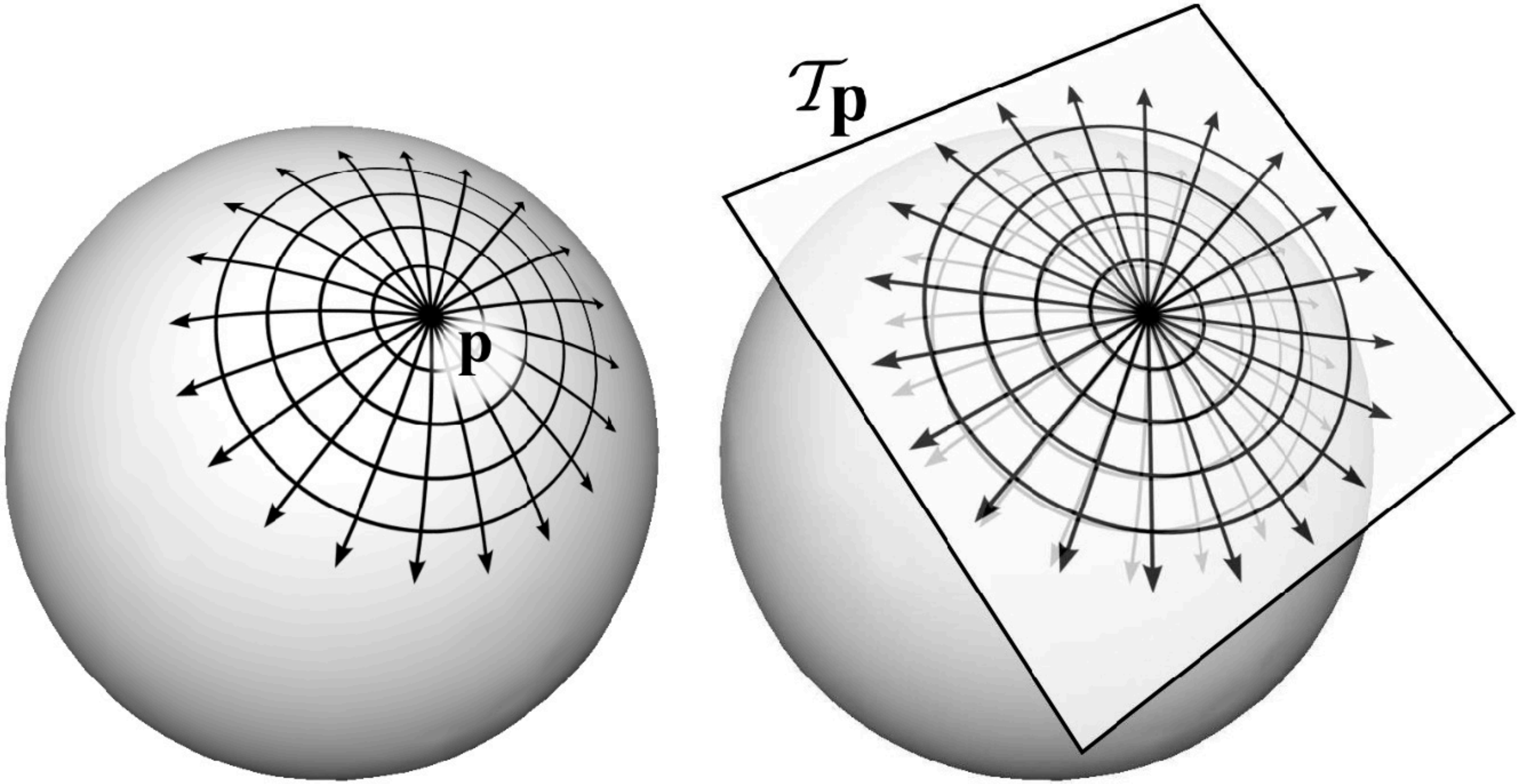


Maron et al. 2017

Local Parameterization

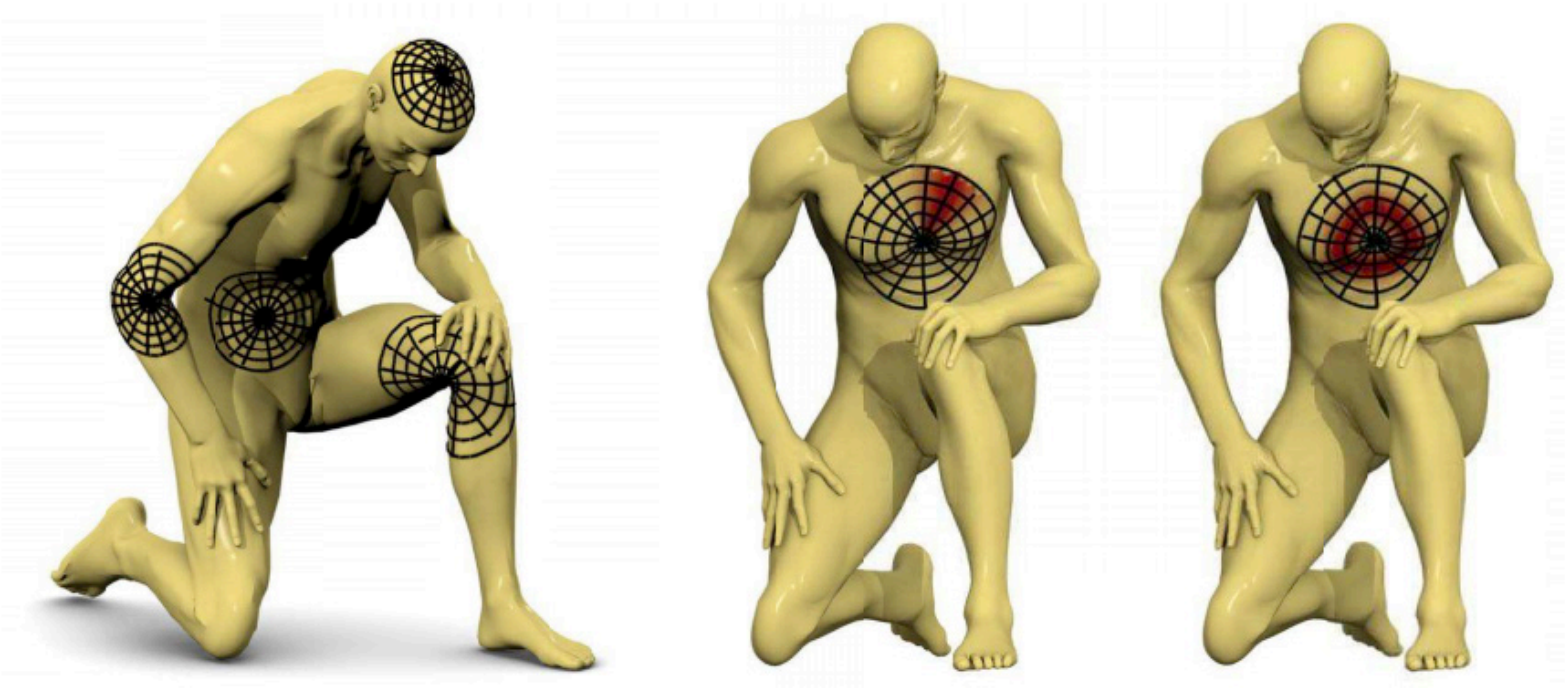


Exponential Maps



Schmidt et al. 2006

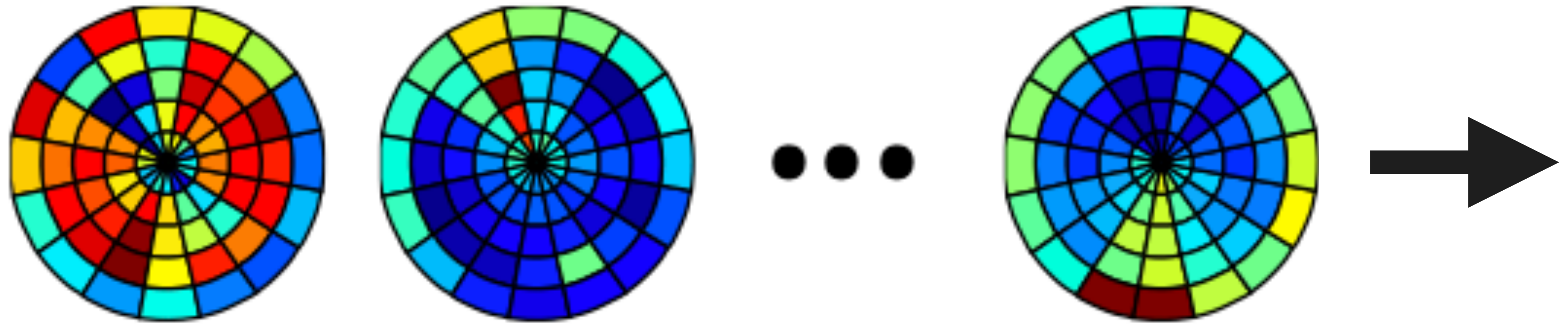
Geodesic CNNs



Masci et al. 2015

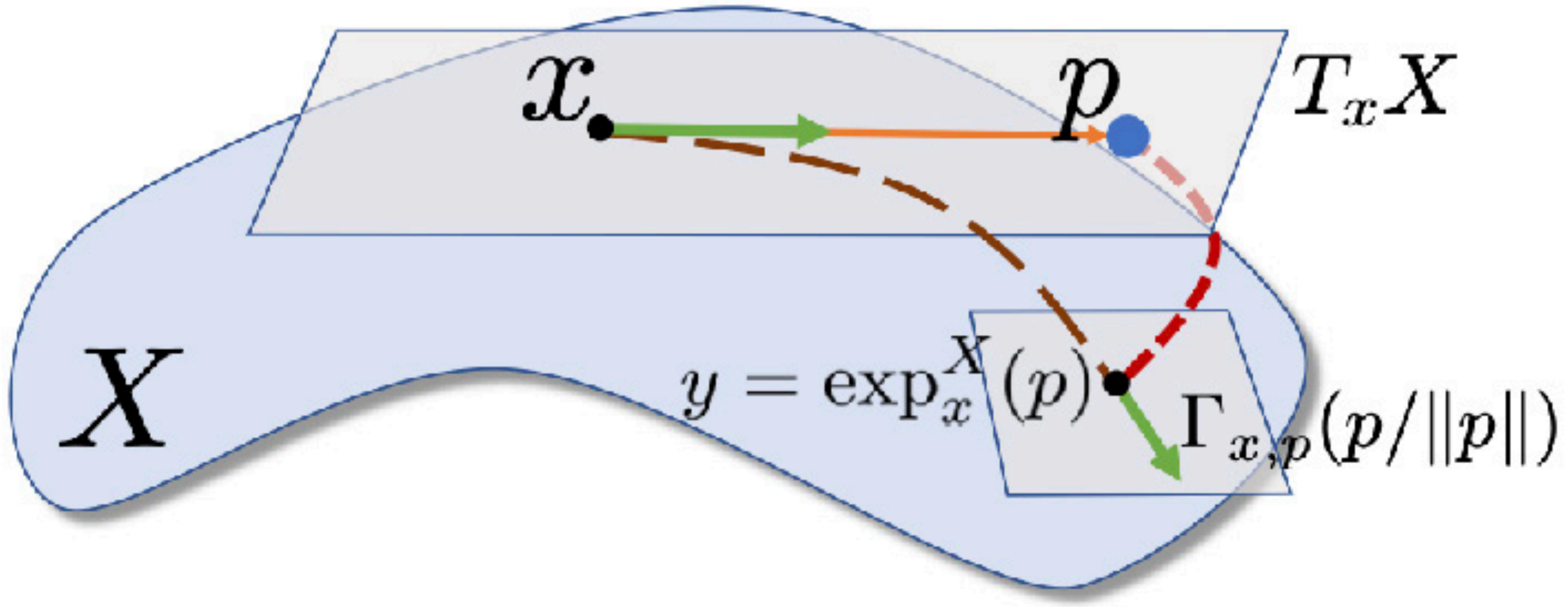
Which direction to use?

Consider all orientations
[Masci et al. 2015]



Max. / Avg.

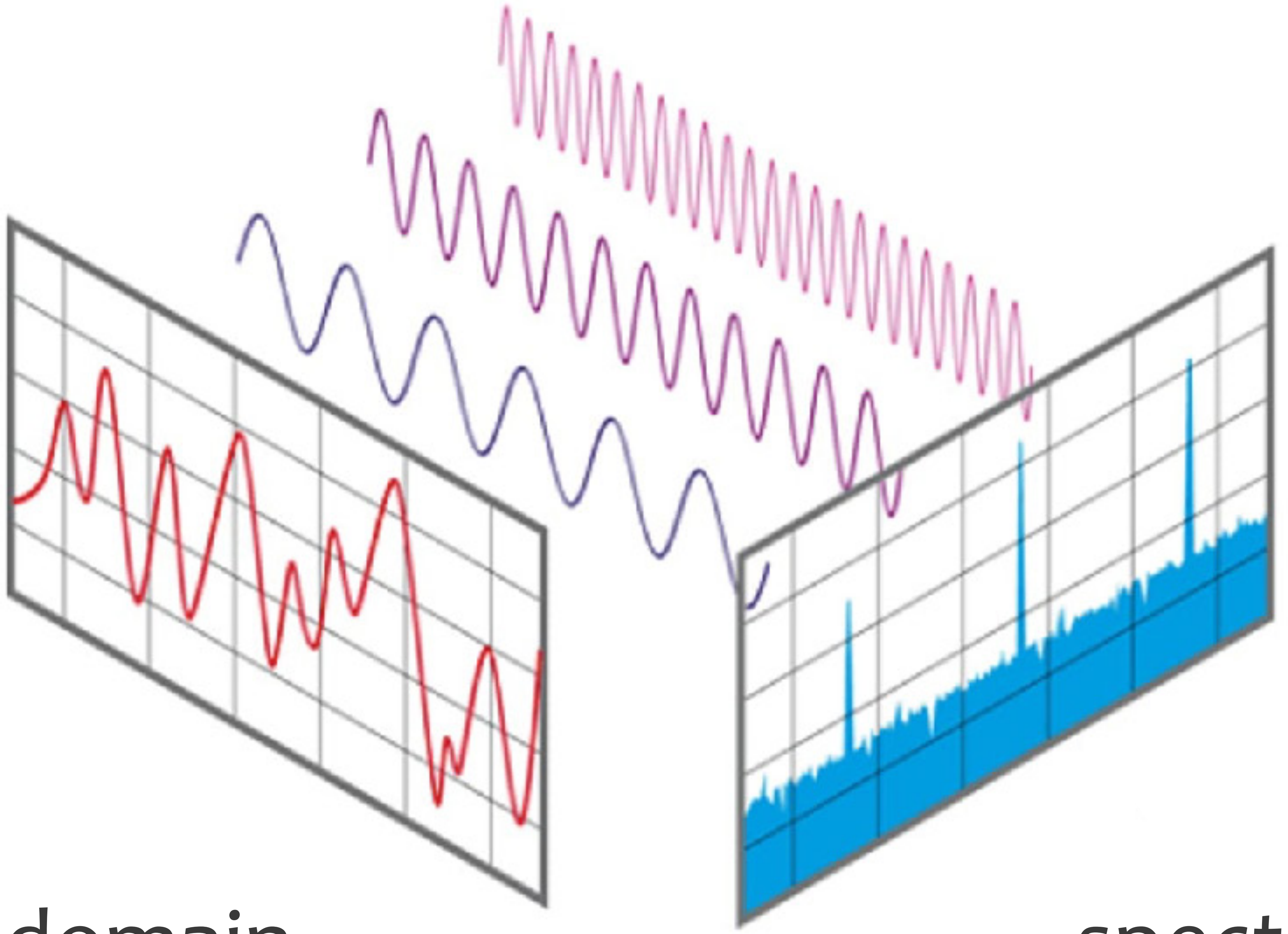
Pick one direction at a time
[Poulenard & Ovsjanikov 2018]



Rotation-Equivariant
[Wiersma et al. 2020]

Spectral Convolution

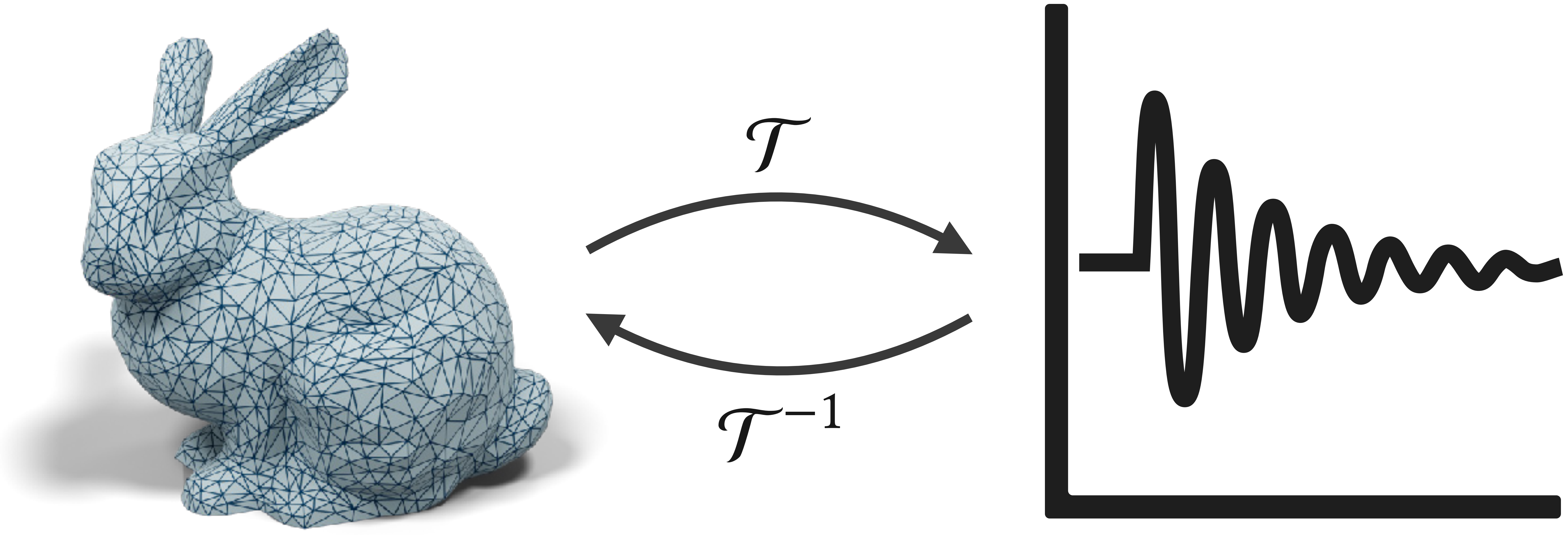
Convolution in the spatial domain is the **point-wise product** in the spectral domain.



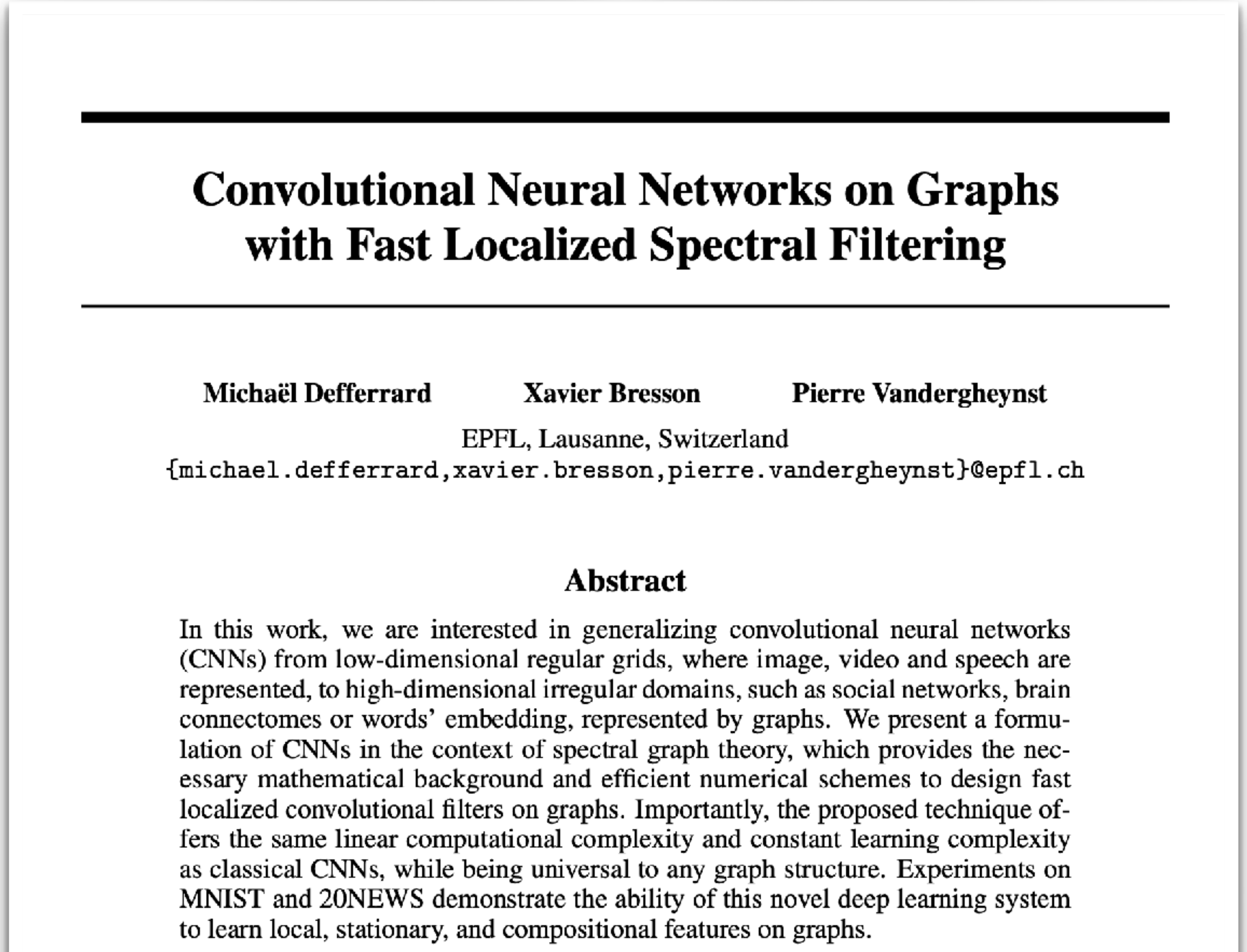
source: wikimedia

Spectral Convolution (e.g., [Defferrard et al. 2016])

$$y = \underbrace{\mathcal{T}^{-1}}_{\text{inverse FT}} \Big(\overbrace{w}^{\text{filter weights}} \odot \underbrace{\mathcal{T}}_{\text{FT}} (\overbrace{x}^{\text{spatial signal}}) \Big)$$



Issues on Efficiency, Localization, Memory

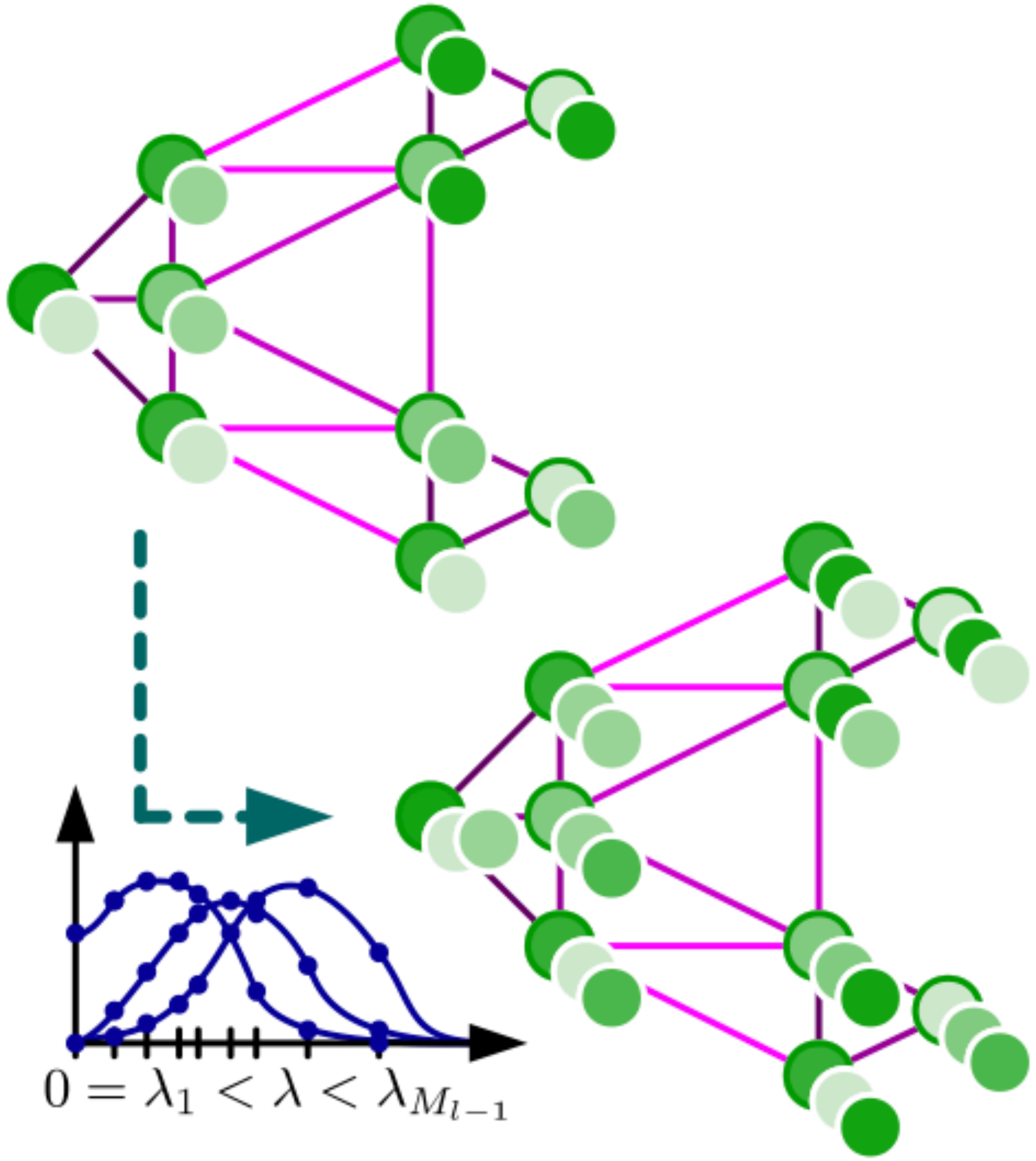


Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Michaël Defferrard **Xavier Bresson** **Pierre Vandergheynst**
EPFL, Lausanne, Switzerland
{michael.defferrard,xavier.bresson,pierre.vandergheynst}@epfl.ch

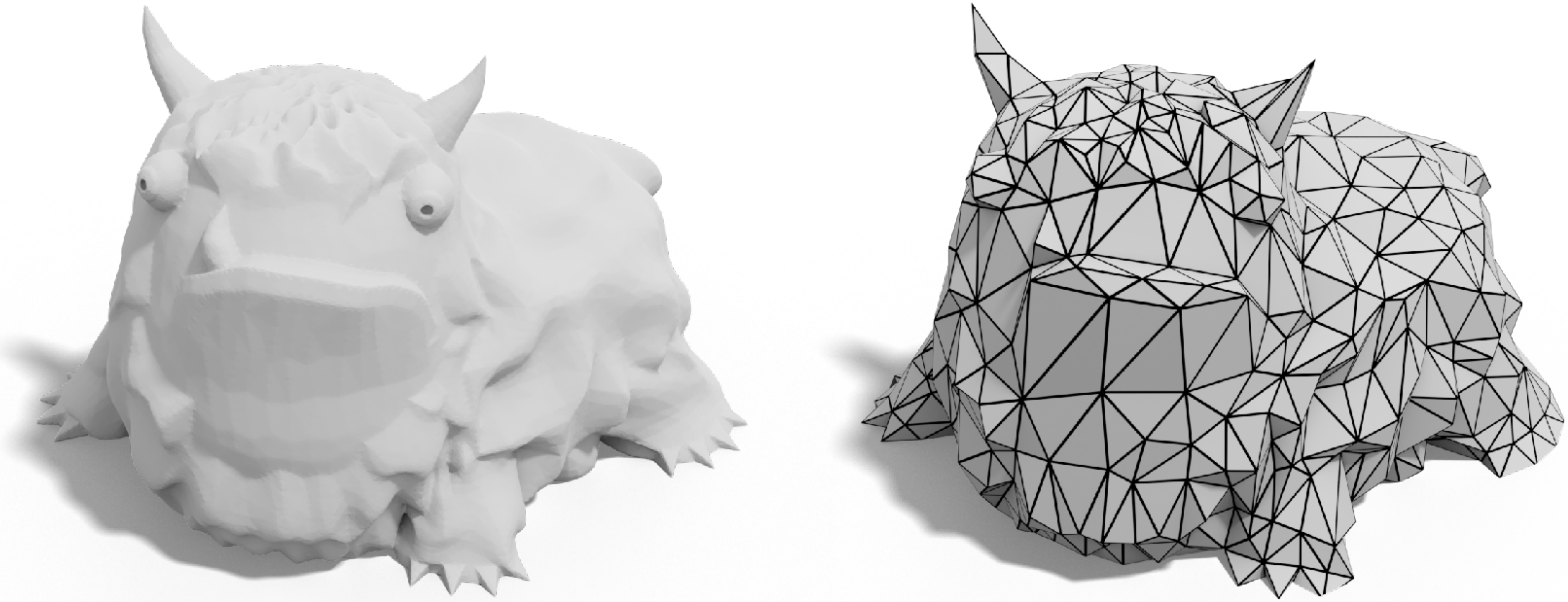
Abstract

In this work, we are interested in generalizing convolutional neural networks (CNNs) from low-dimensional regular grids, where image, video and speech are represented, to high-dimensional irregular domains, such as social networks, brain connectomes or words' embedding, represented by graphs. We present a formulation of CNNs in the context of spectral graph theory, which provides the necessary mathematical background and efficient numerical schemes to design fast localized convolutional filters on graphs. Importantly, the proposed technique offers the same linear computational complexity and constant learning complexity as classical CNNs, while being universal to any graph structure. Experiments on MNIST and 20NEWS demonstrate the ability of this novel deep learning system to learn local, stationary, and compositional features on graphs.



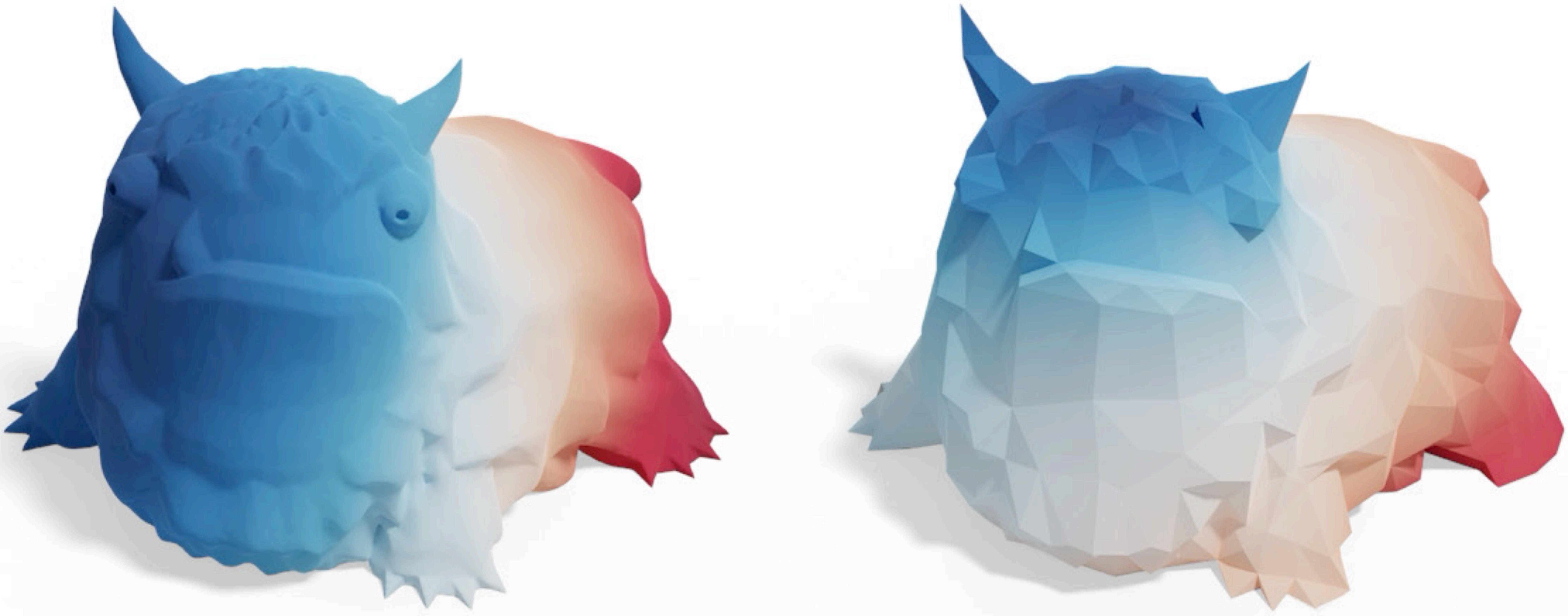
Do not generalize to other shapes

(even the same shape with a different connectivity)



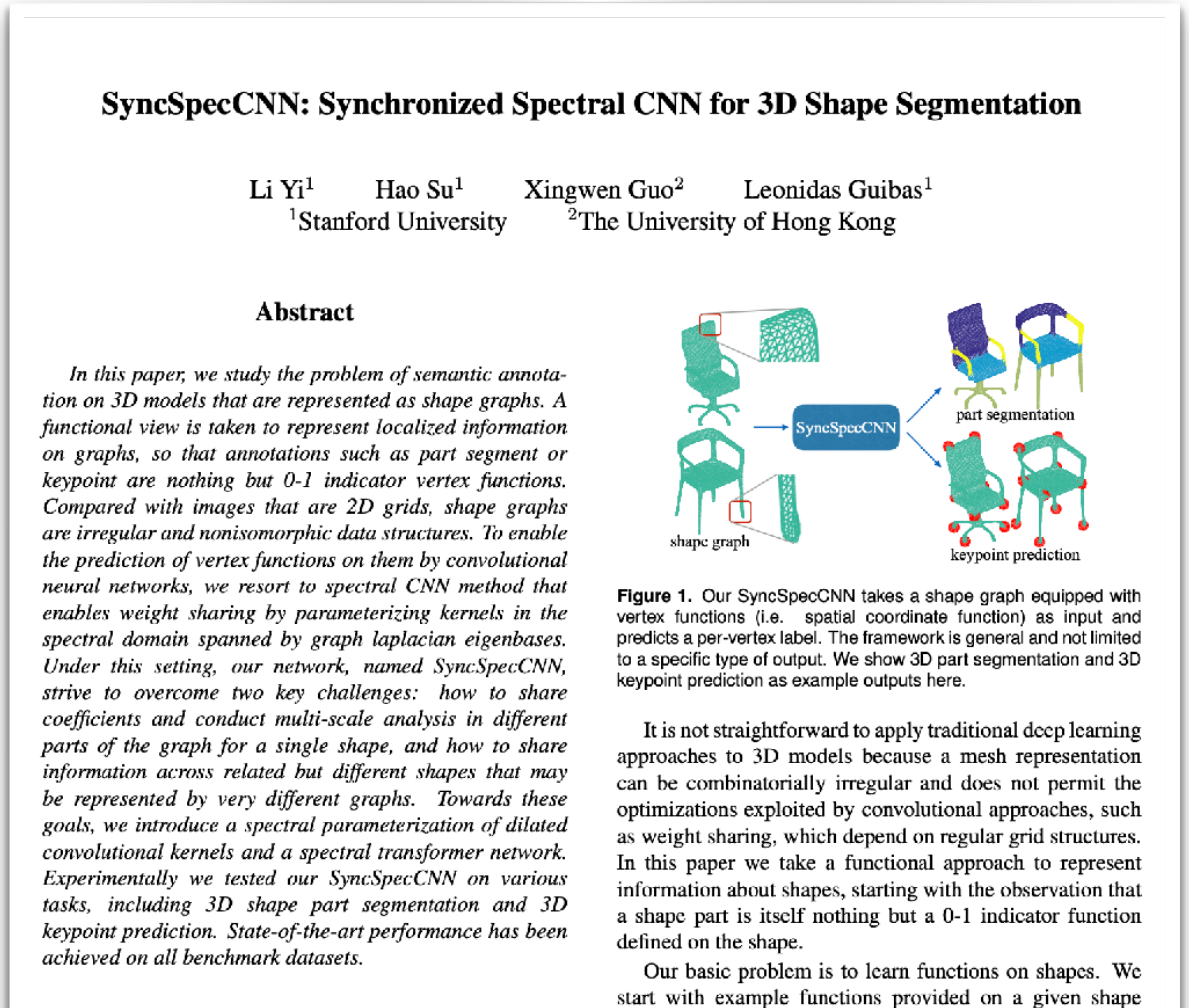
Do not generalize to other shapes

(even the same shape with a different connectivity)



different Fourier bases

Synchronizing Spectral Spaces



SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation

Li Yi[1] Hao Su[1] Xingwen Guo[2] Leonidas Guibas[1]
[1]Stanford University [2]The University of Hong Kong

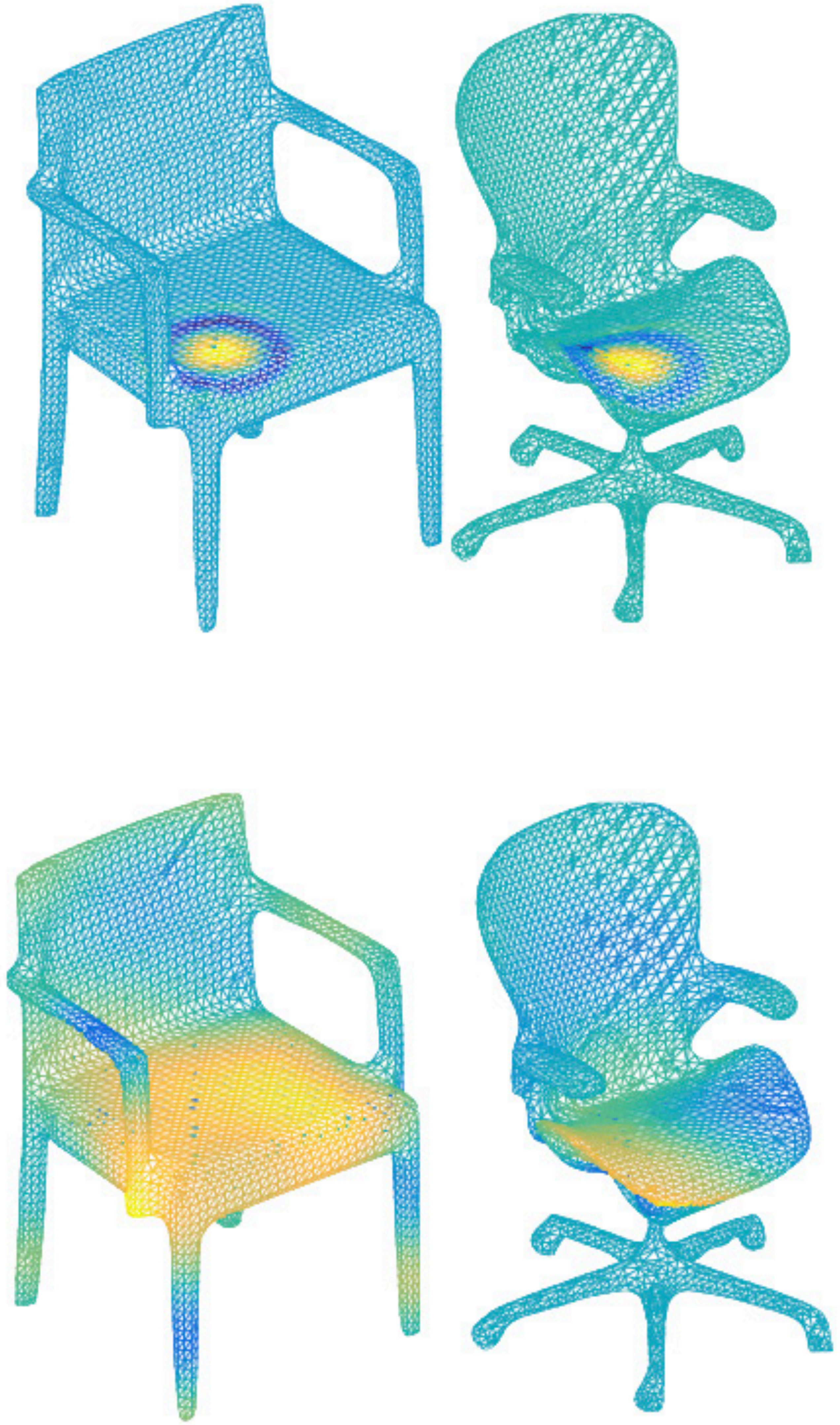
Abstract

In this paper, we study the problem of semantic annotation on 3D models that are represented as shape graphs. A functional view is taken to represent localized information on graphs, so that annotations such as part segment or keypoint are nothing but 0-1 indicator vertex functions. Compared with images that are 2D grids, shape graphs are irregular and nonisomorphic data structures. To enable the prediction of vertex functions on them by convolutional neural networks, we resort to spectral CNN method that enables weight sharing by parameterizing kernels in the spectral domain spanned by graph laplacian eigenbases. Under this setting, our network, named SyncSpecCNN, strive to overcome two key challenges: how to share coefficients and conduct multi-scale analysis in different parts of the graph for a single shape, and how to share information across related but different shapes that may be represented by very different graphs. Towards these goals, we introduce a spectral parameterization of dilated convolutional kernels and a spectral transformer network. Experimentally we tested our SyncSpecCNN on various tasks, including 3D shape part segmentation and 3D keypoint prediction. State-of-the-art performance has been achieved on all benchmark datasets.

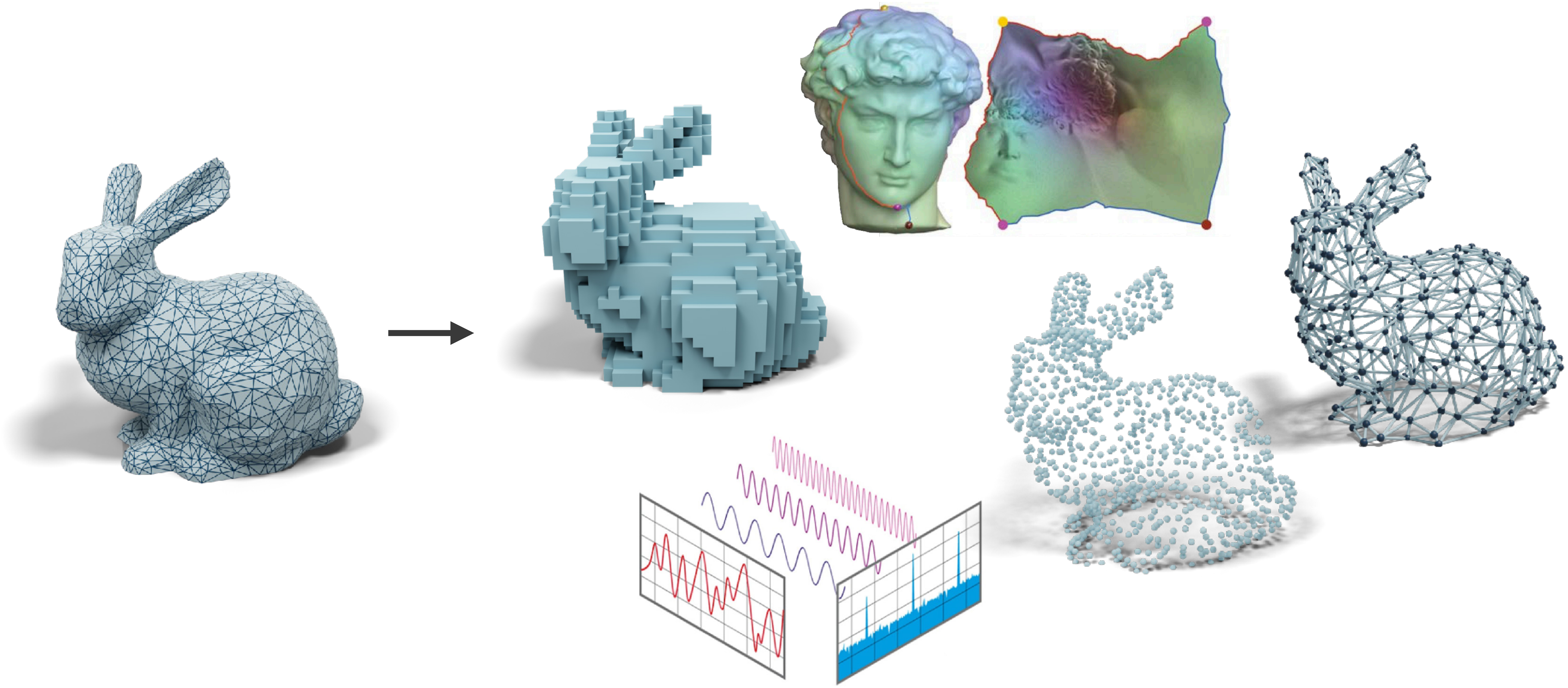
Figure 1. Our SyncSpecCNN takes a shape graph equipped with vertex functions (i.e. spatial coordinate function) as input and predicts a per-vertex label. The framework is general and not limited to a specific type of output. We show 3D part segmentation and 3D keypoint prediction as example outputs here.

It is not straightforward to apply traditional deep learning approaches to 3D models because a mesh representation can be combinatorially irregular and does not permit the optimizations exploited by convolutional approaches, such as weight sharing, which depend on regular grid structures. In this paper we take a functional approach to represent information about shapes, starting with the observation that a shape part is itself nothing but a 0-1 indicator function defined on the shape.

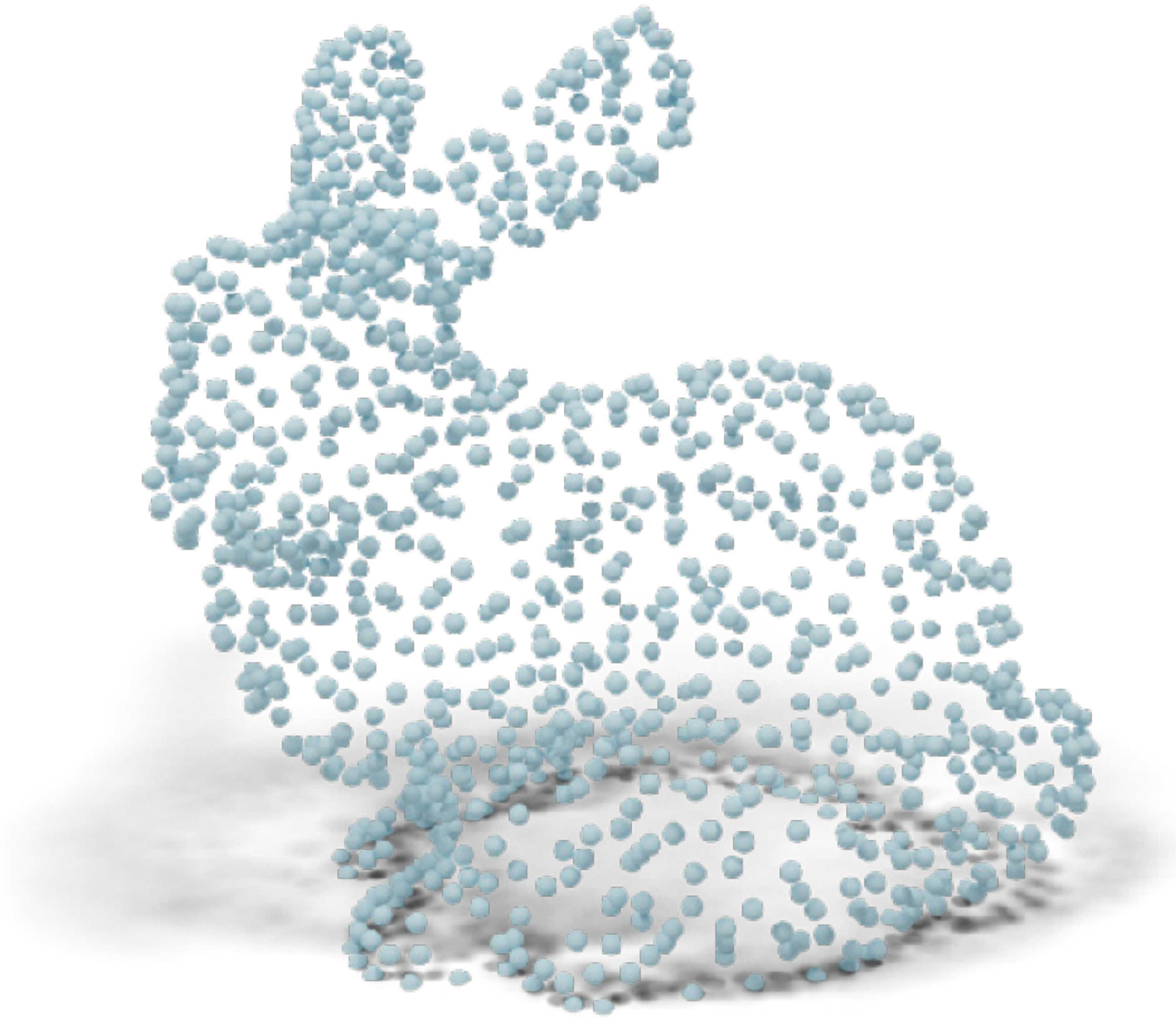
Our basic problem is to learn functions on shapes. We start with example functions provided on a given shape



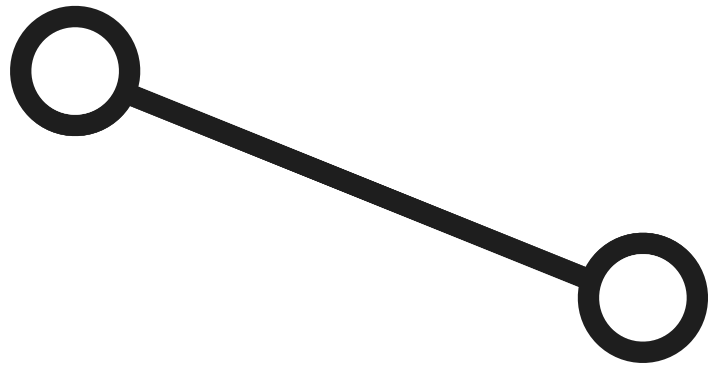
Different representations



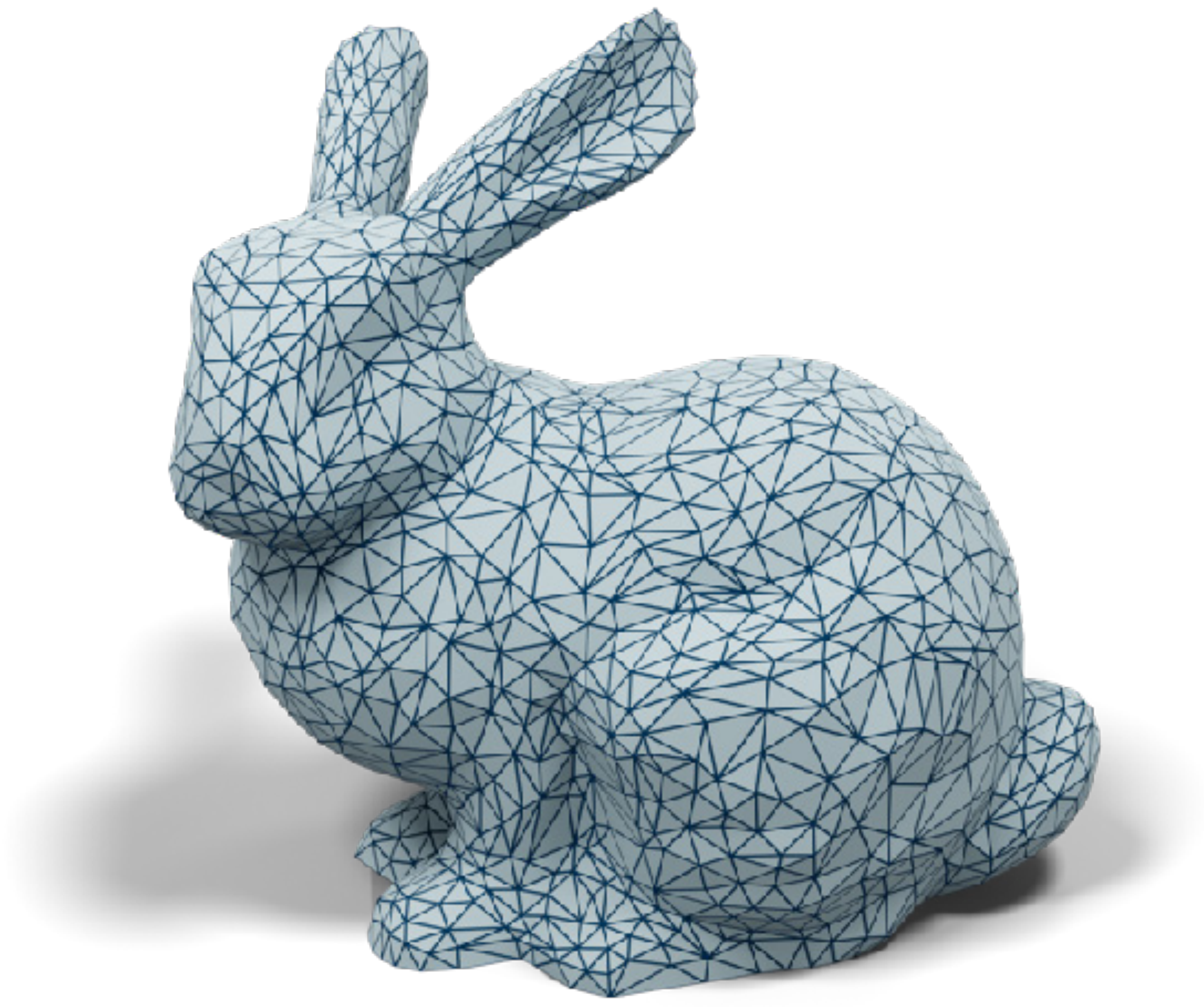
Why not point cloud convolution?



+

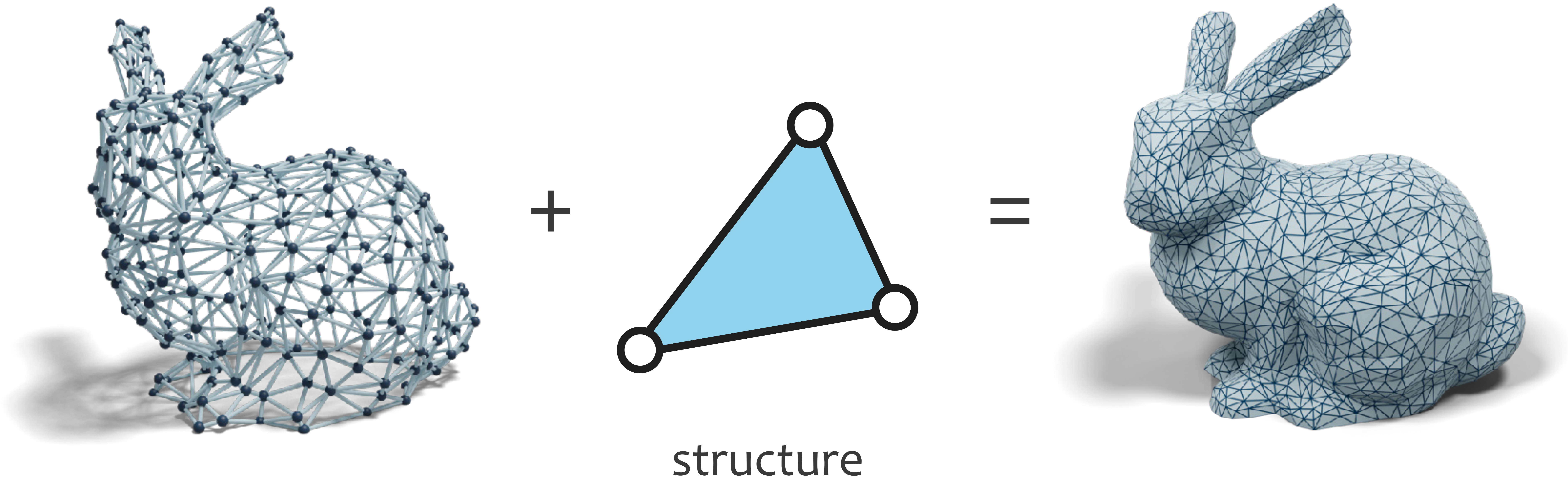


=

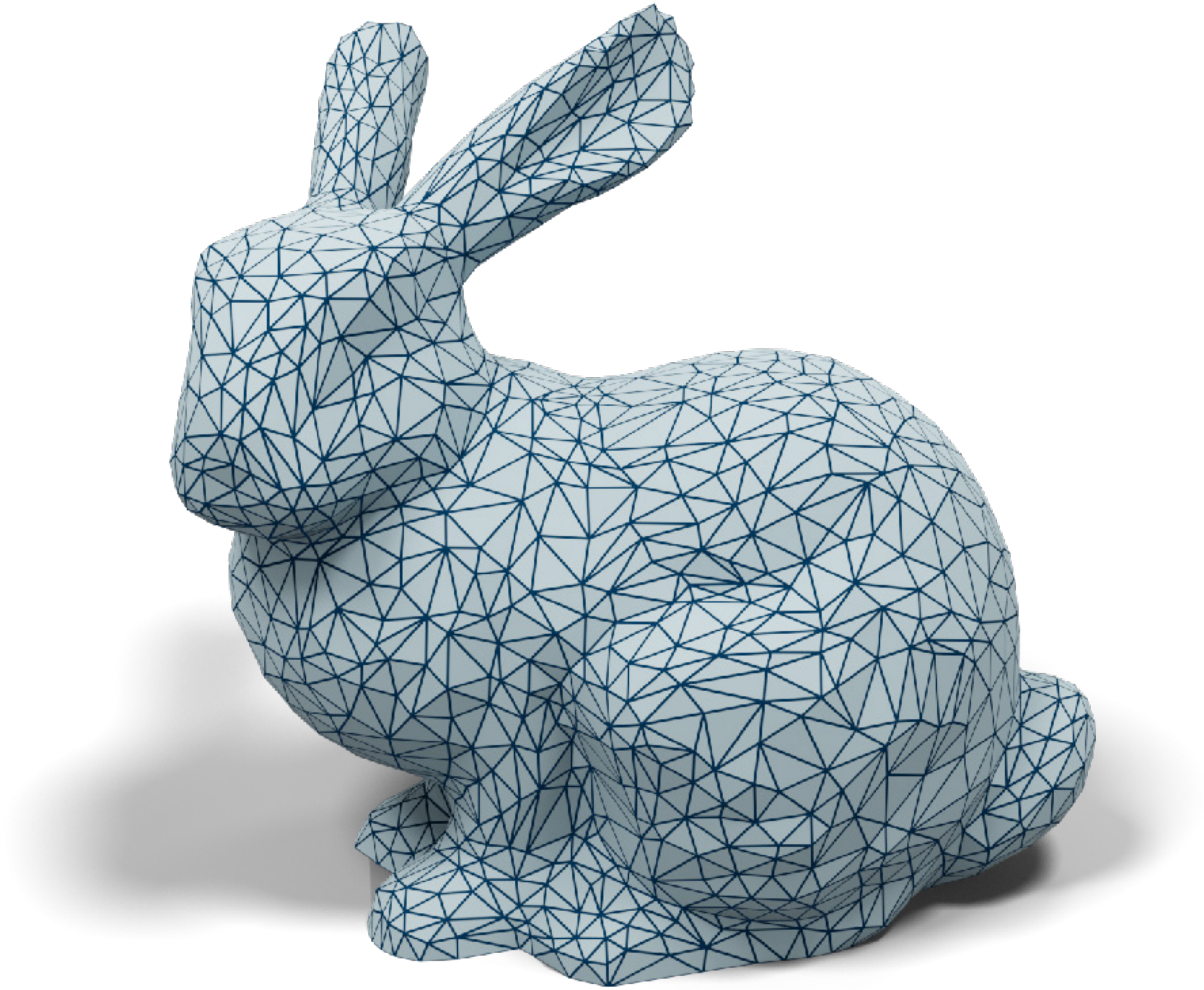


connectivity

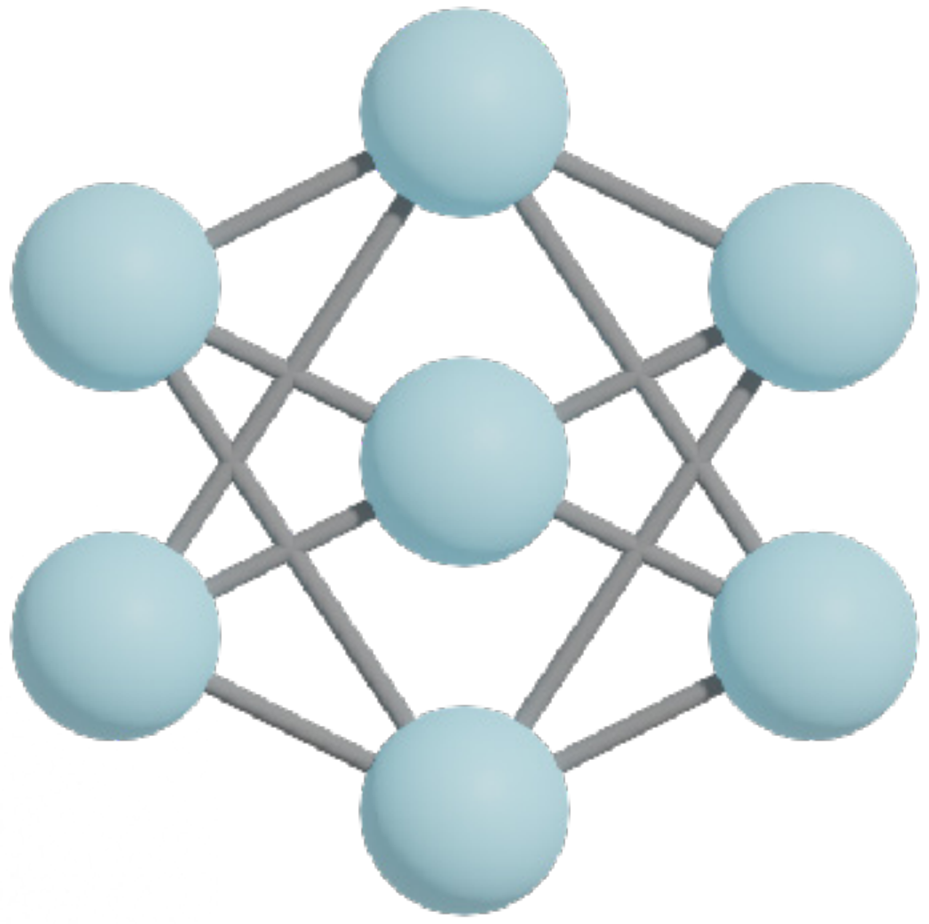
Why not graph convolution?



Surface Meshes



Neural Networks on Meshes



Example Outputs

Global shape descriptor

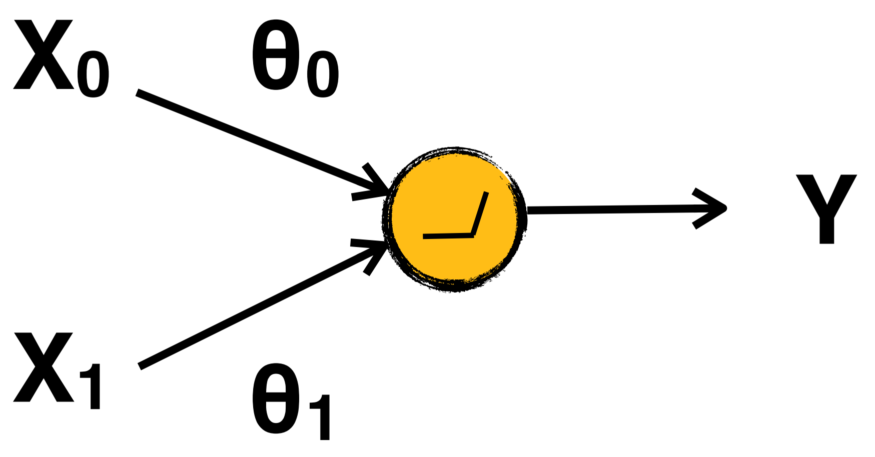
Probability to collapse an edge

Displacement per vertex

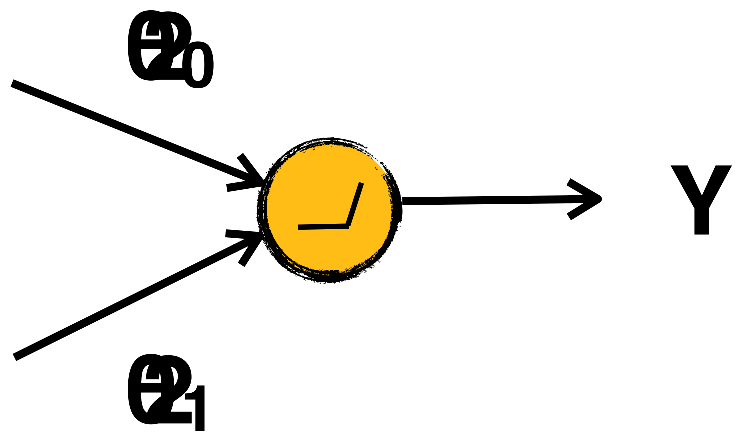
Segmentation label per-face

⋮

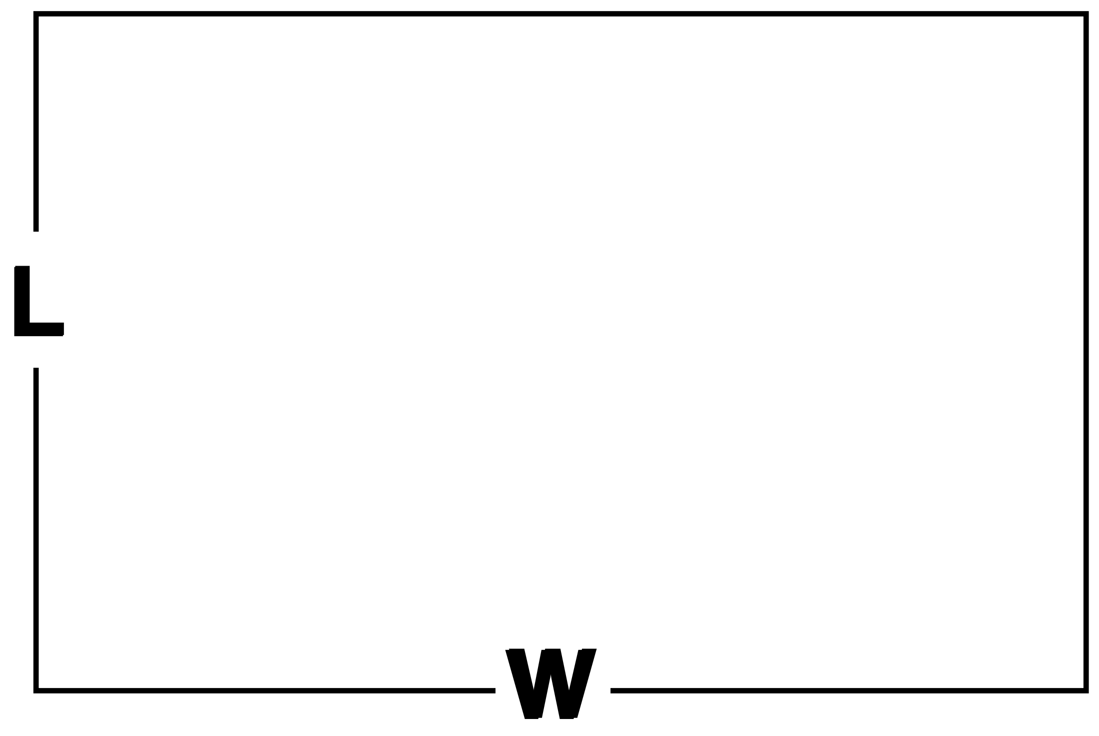
Neuron



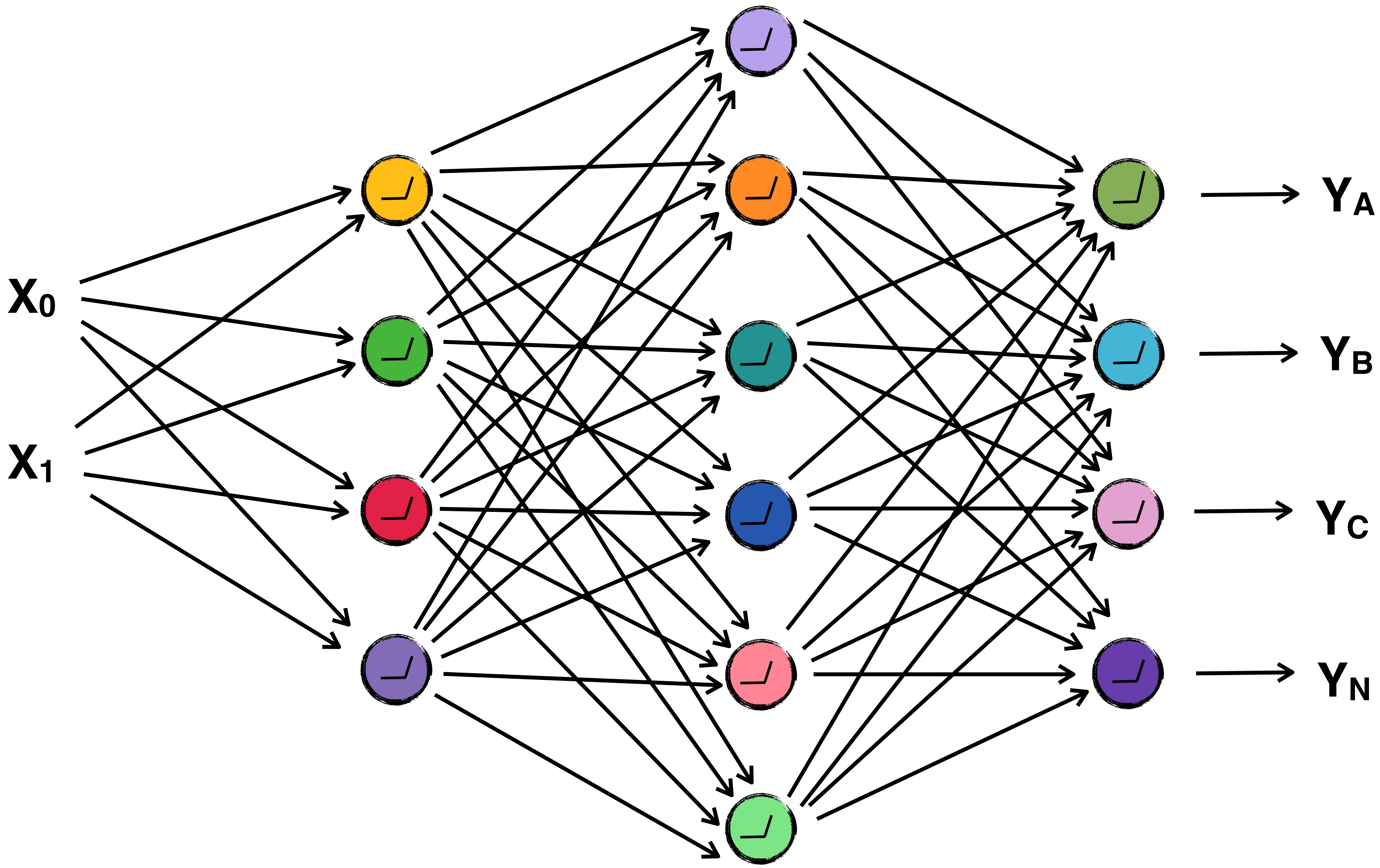
Neuron



Y = ReLU(2×L + 2×W)

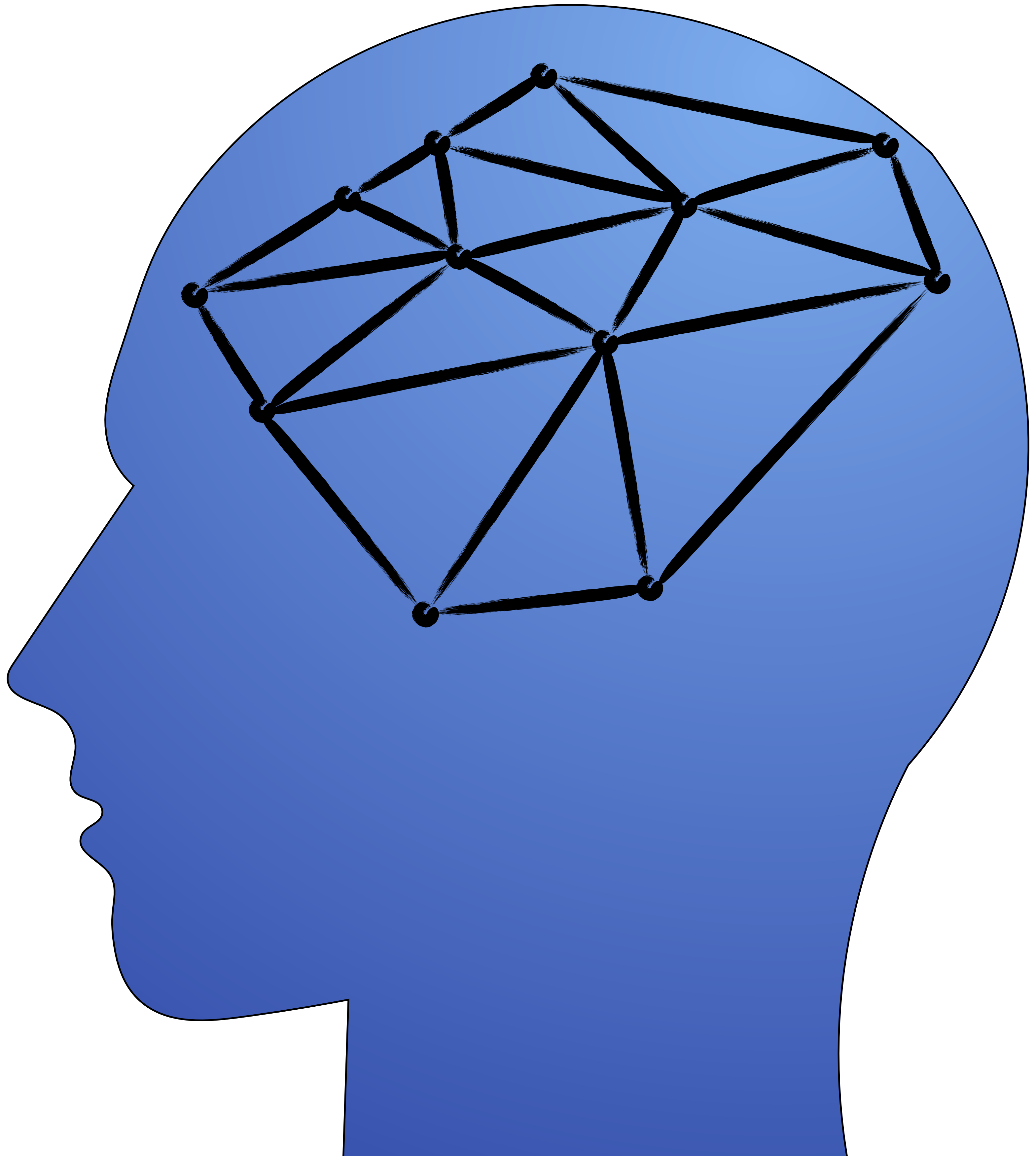


Fully-connected



Inductive Bias

The set of assumptions that we encode into our network, which make it better suited for the task

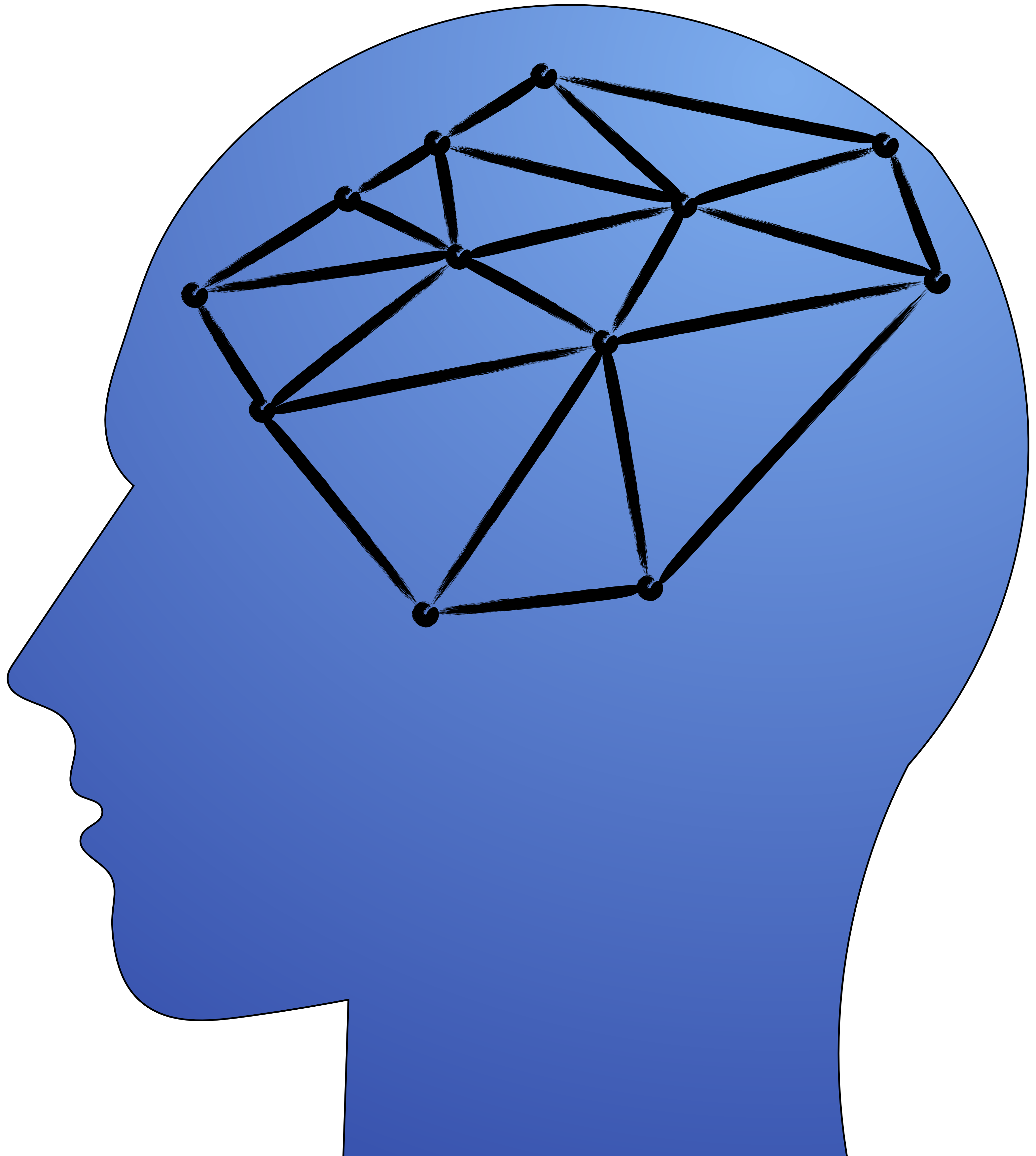


Good

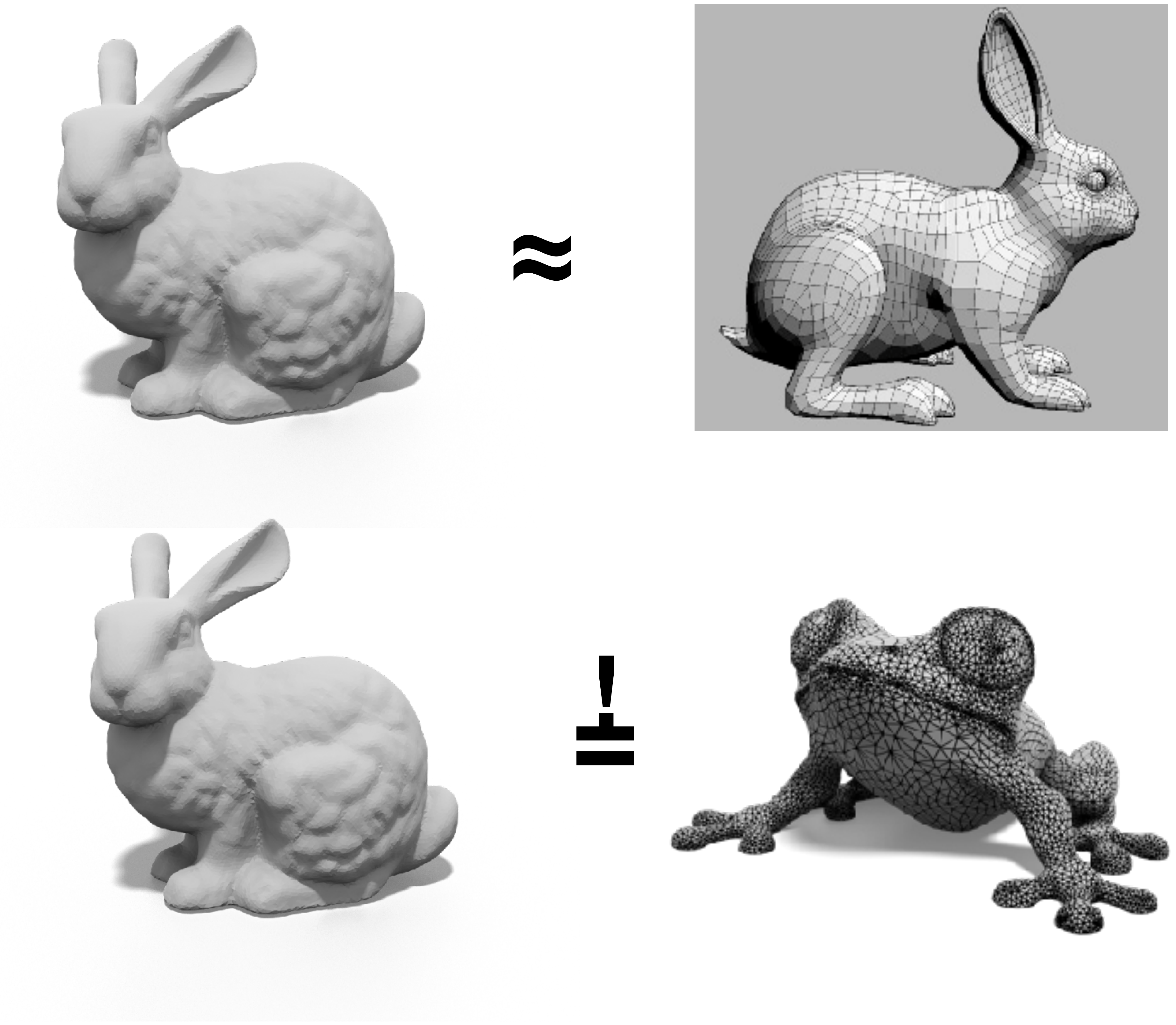
Inductive Bias

↓

Robust to irrelevant variations of the input

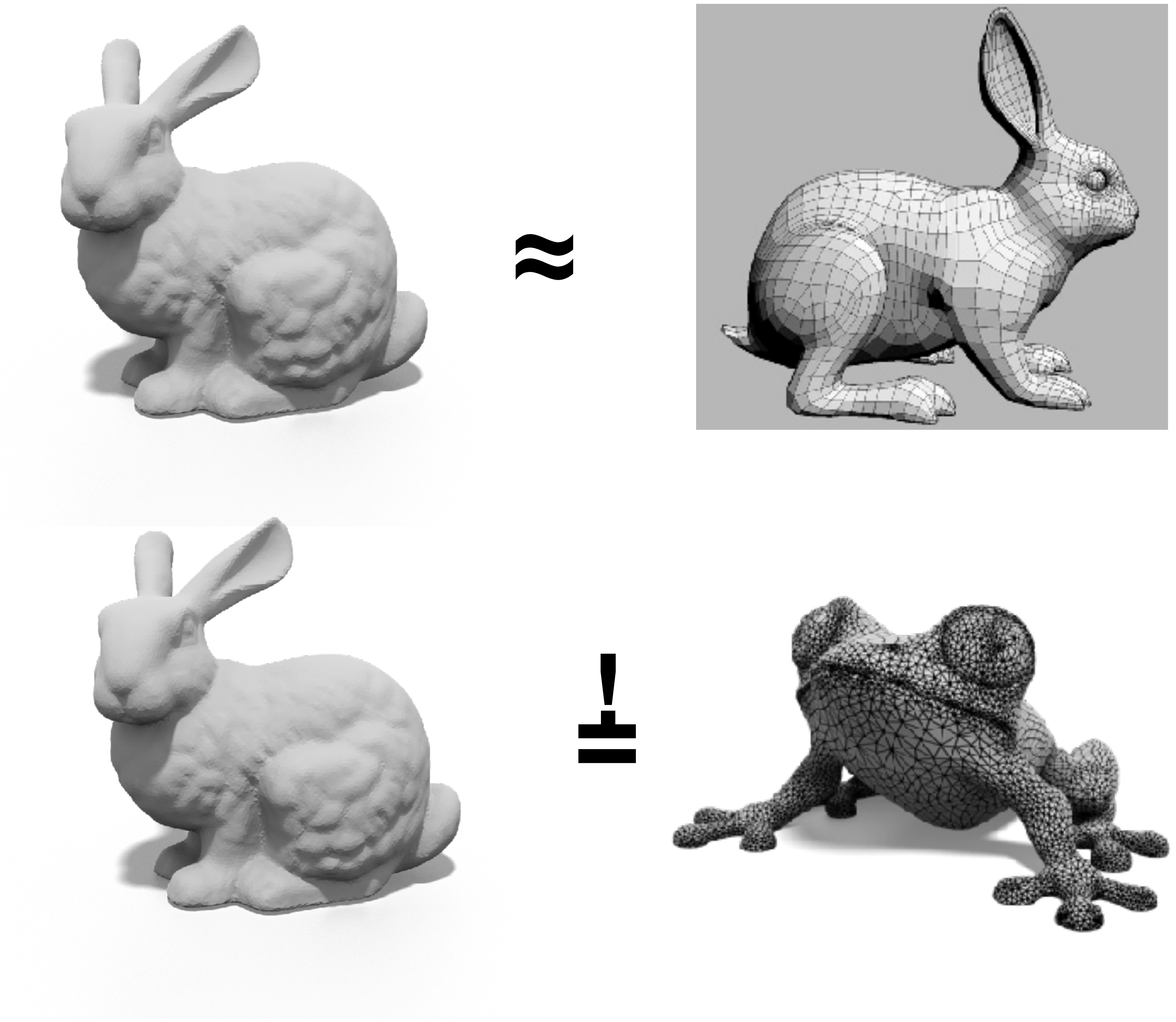


Local tasks



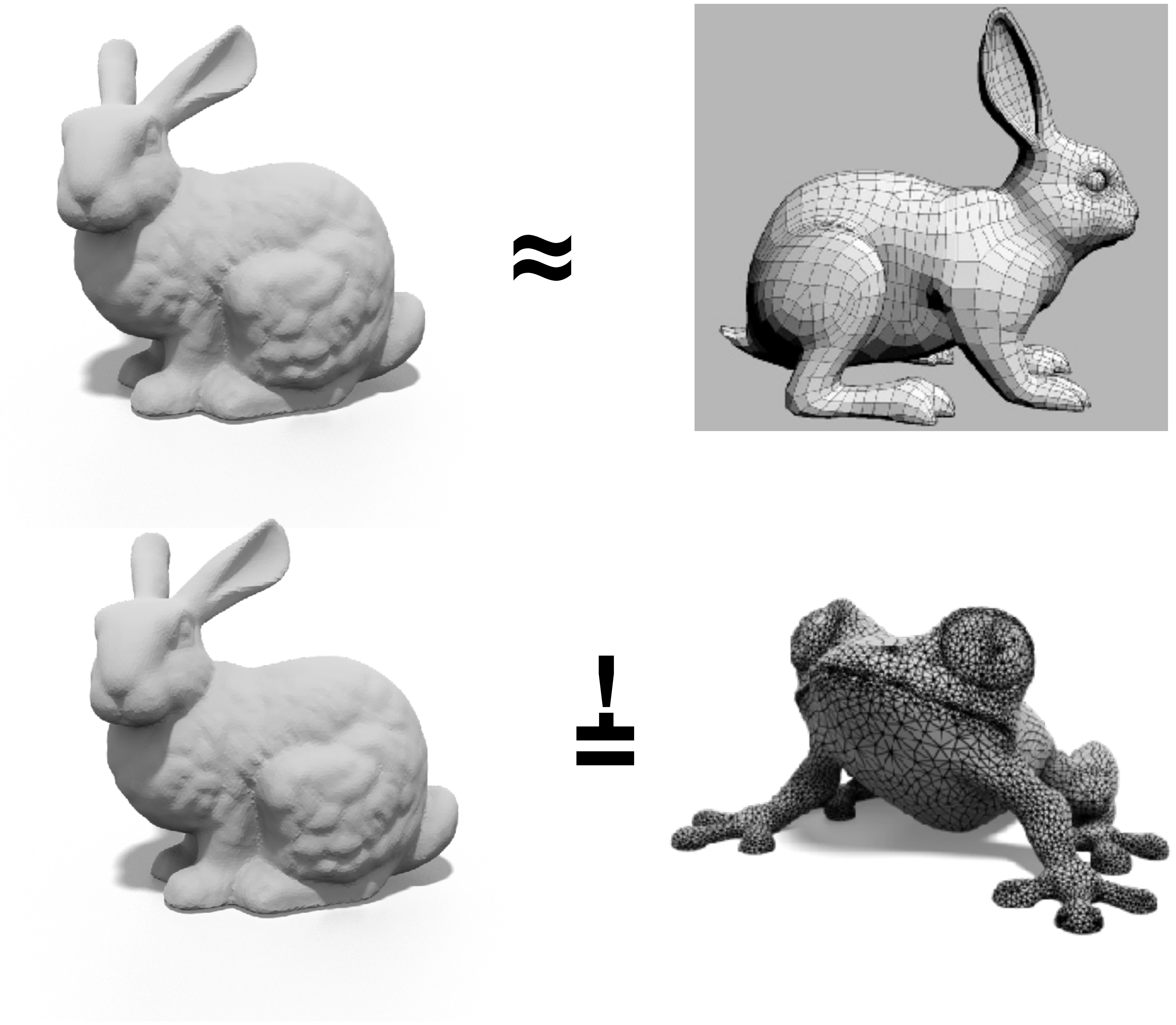
Global tasks

Local tasks



Global tasks

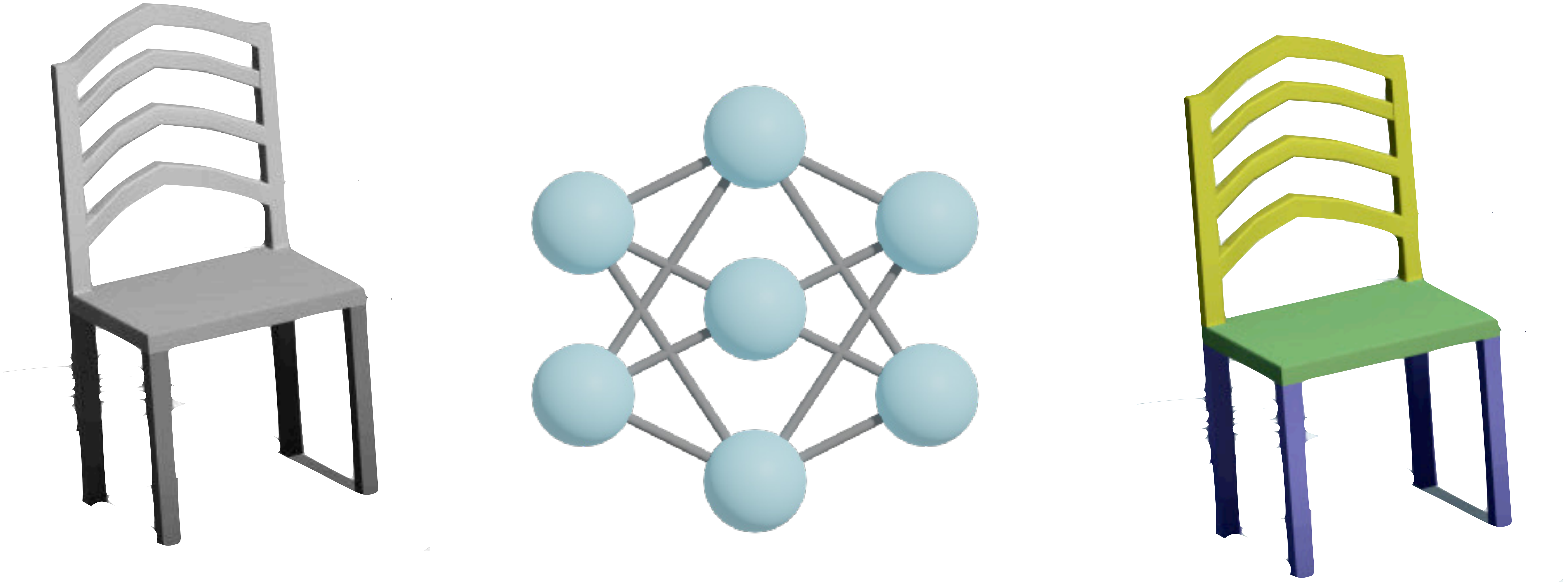
Local tasks



Global tasks

Local task

predict values per mesh element

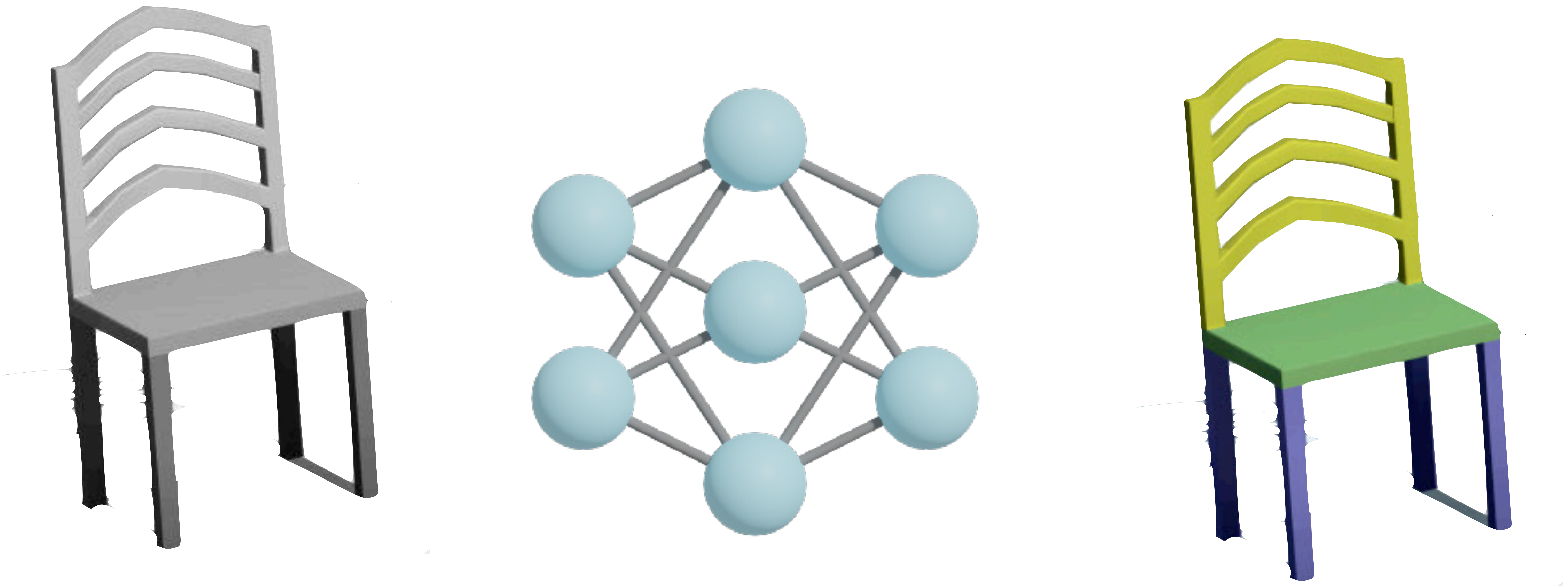


Local task

predict values per mesh element

Faces

f_0
f_1
f_2
f_3
f_4
$\vdots$
f_N



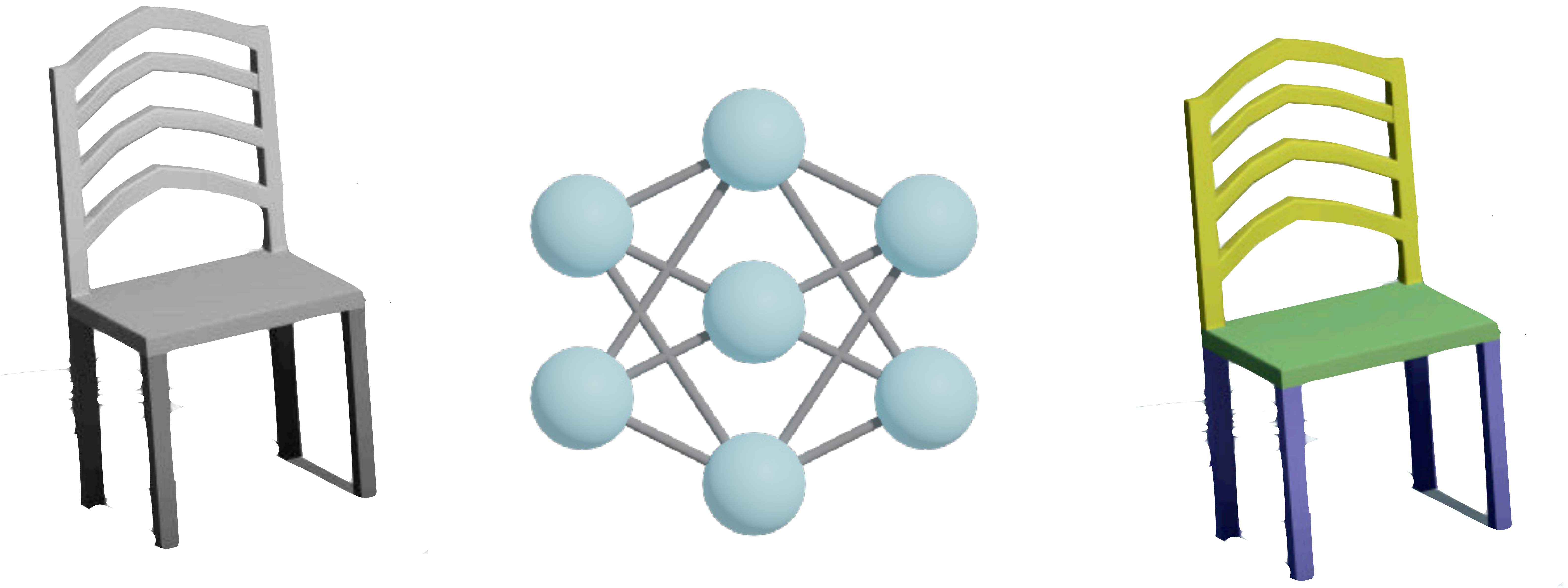
seat
legs
back
seat
legs
⋮
seat

Local task

predict values per mesh element

Faces

f_0
f_1
f_2
f_3
f_4
⋮
f_N

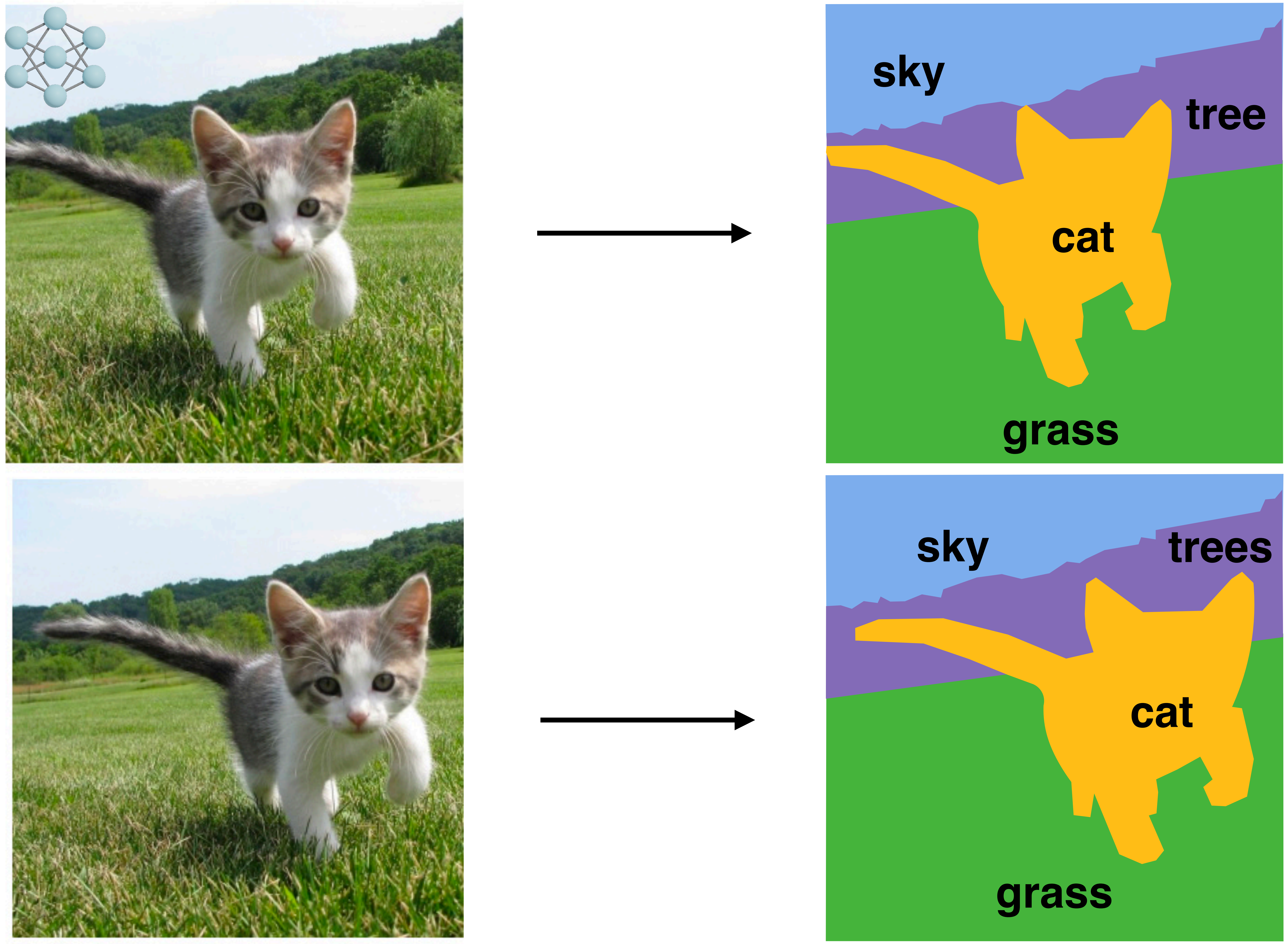


seat
legs
back
seat
legs
⋮
seat

Fully connected network not suitable for this task

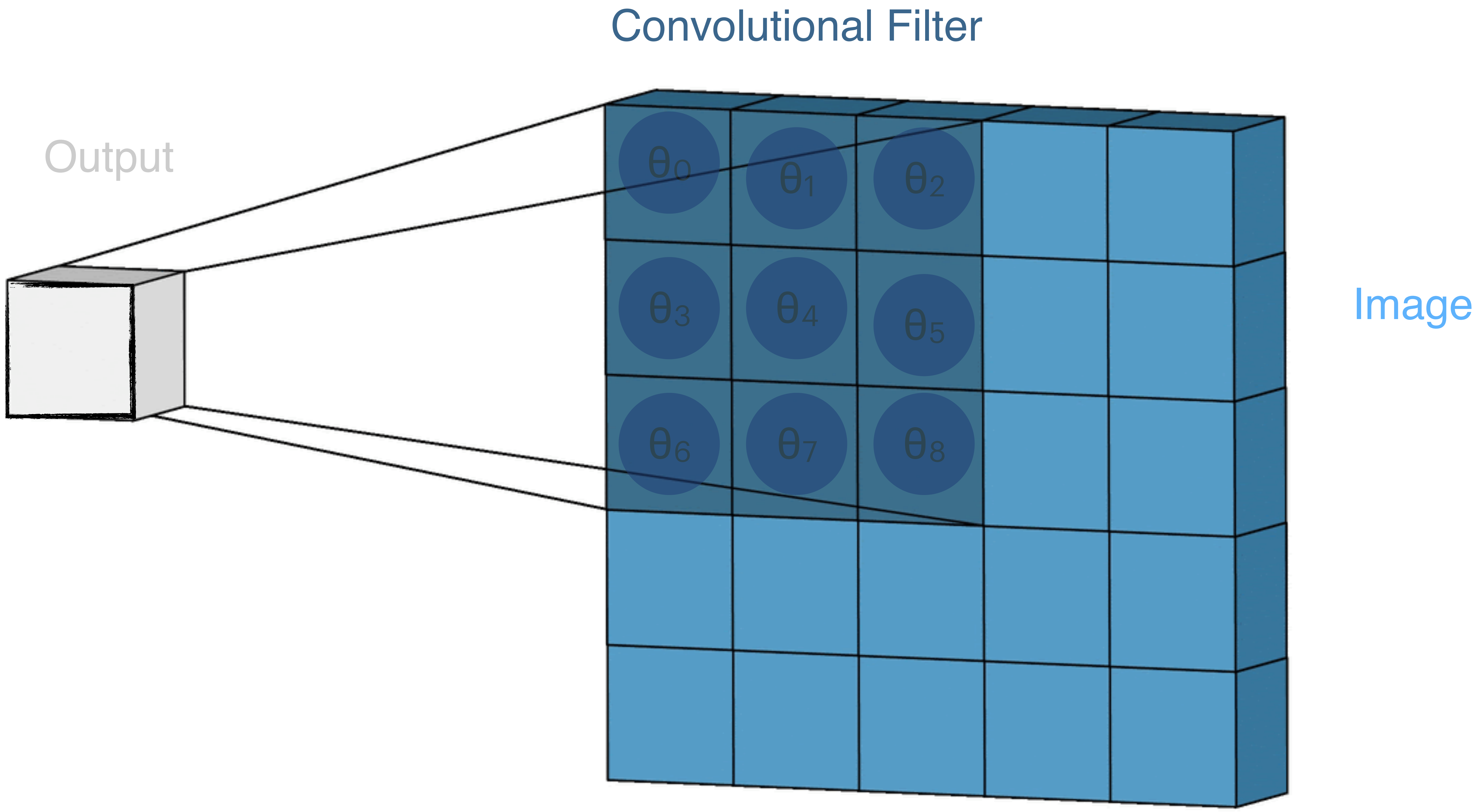
Inspiration: image segmentation

Shared weights are a good inductive bias

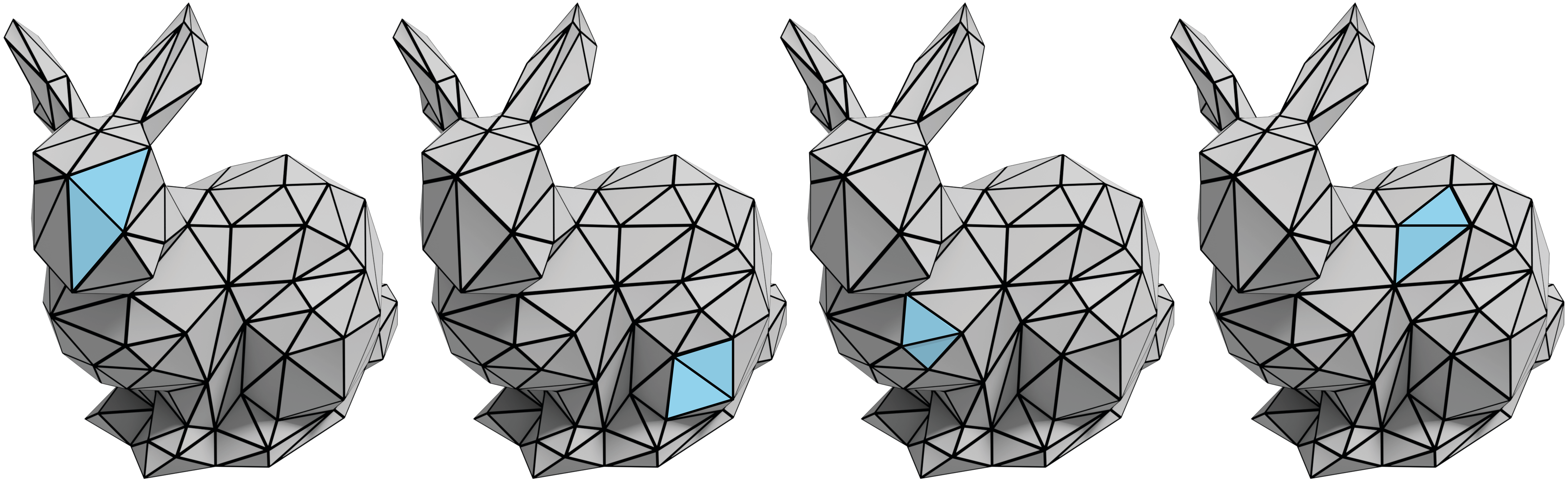


Convolution

Shared-weights

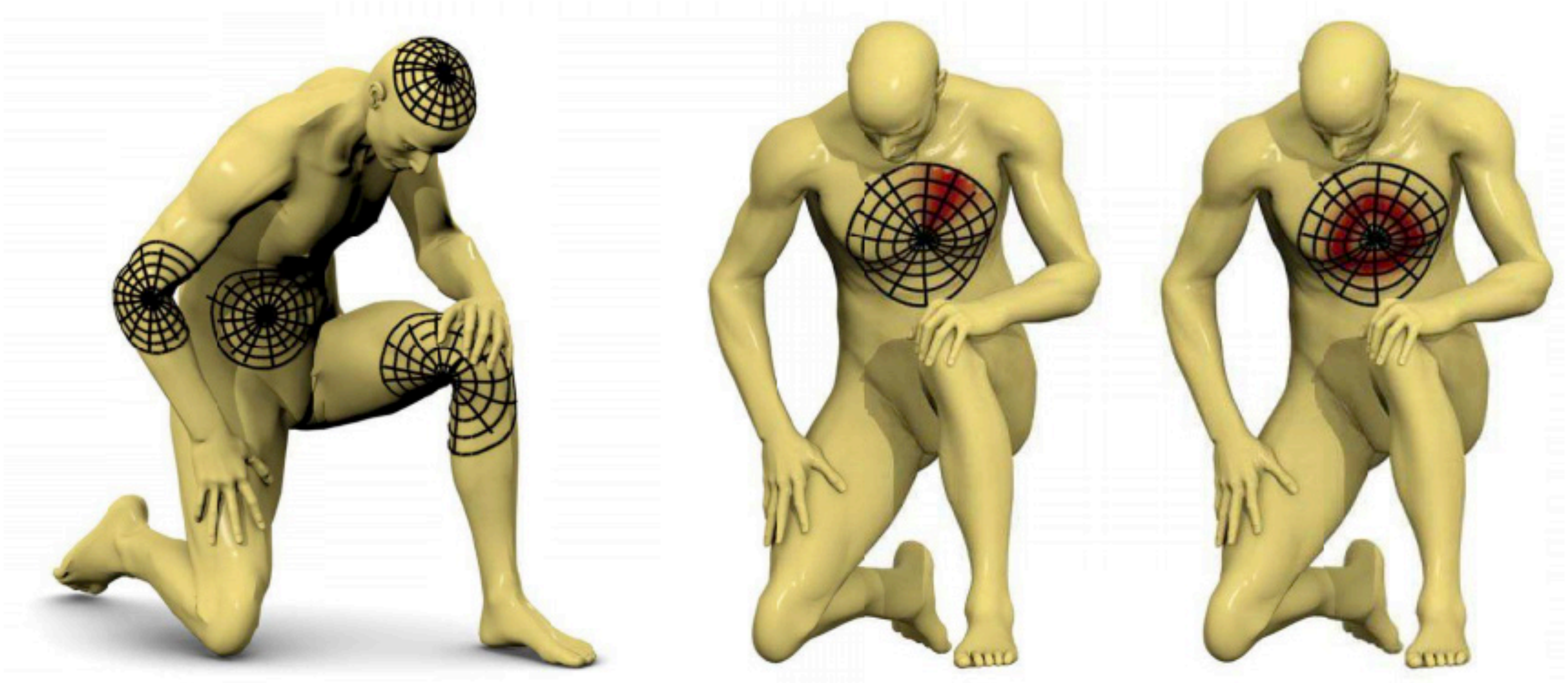


Convolutions on meshes



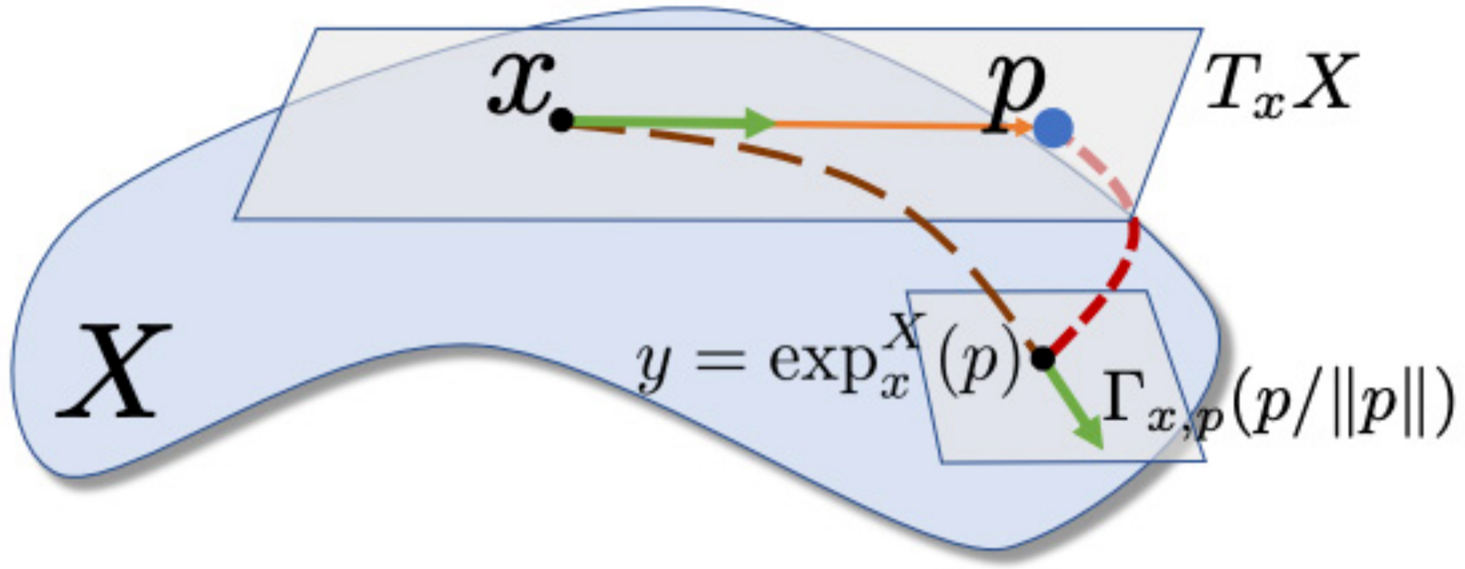
Learn over intrinsic patches

local parameterizations

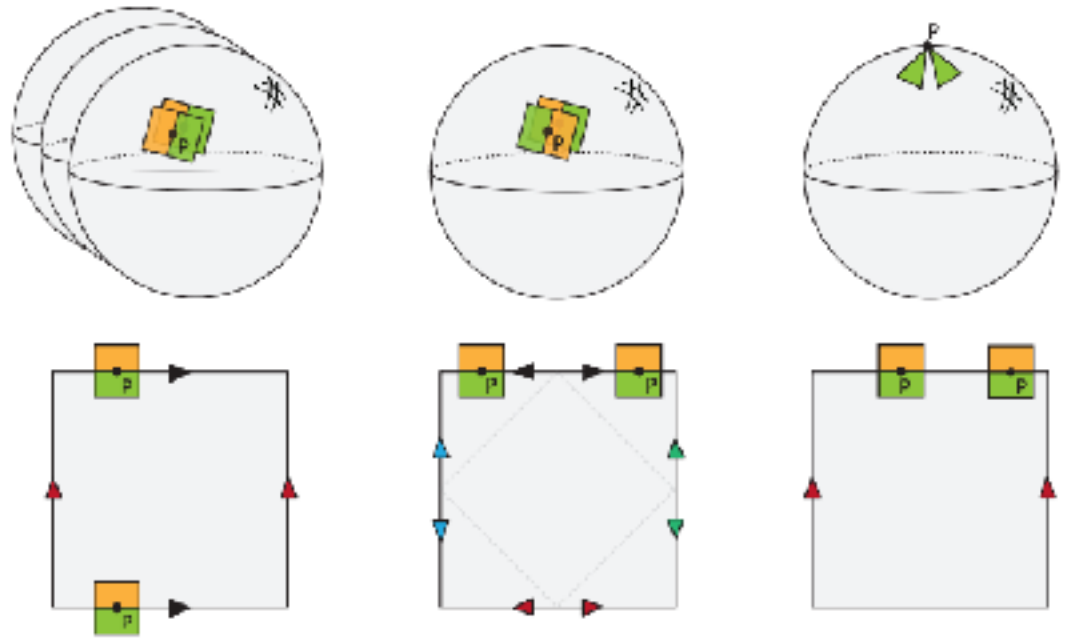


GCNN Masci et. al 2015

Intrinsic Techniques



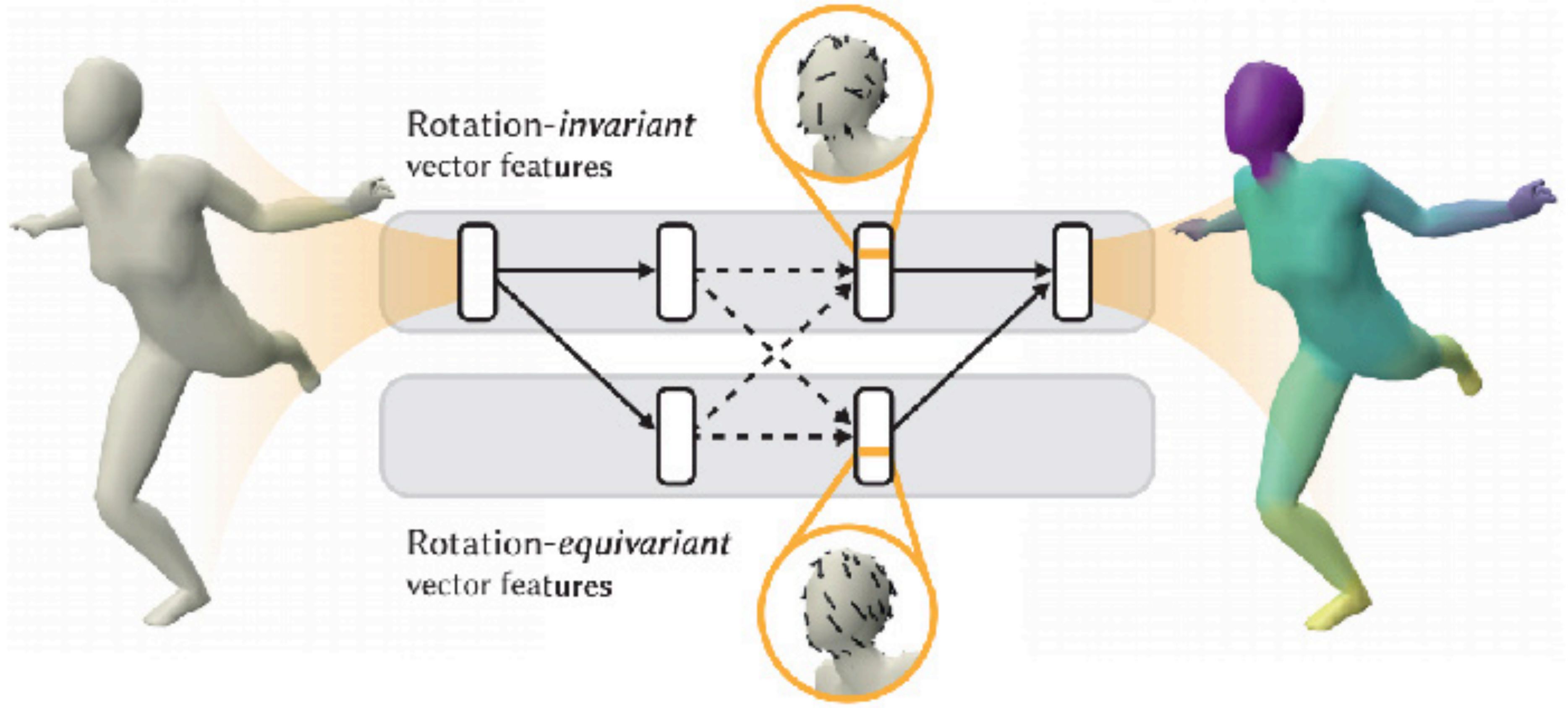
MDGCNN. Poulenard & Ovsjanikov [SIGGRAPH Asia 2018]



Surface Networks via General Covers. Haim et. al [ICCV 2019]

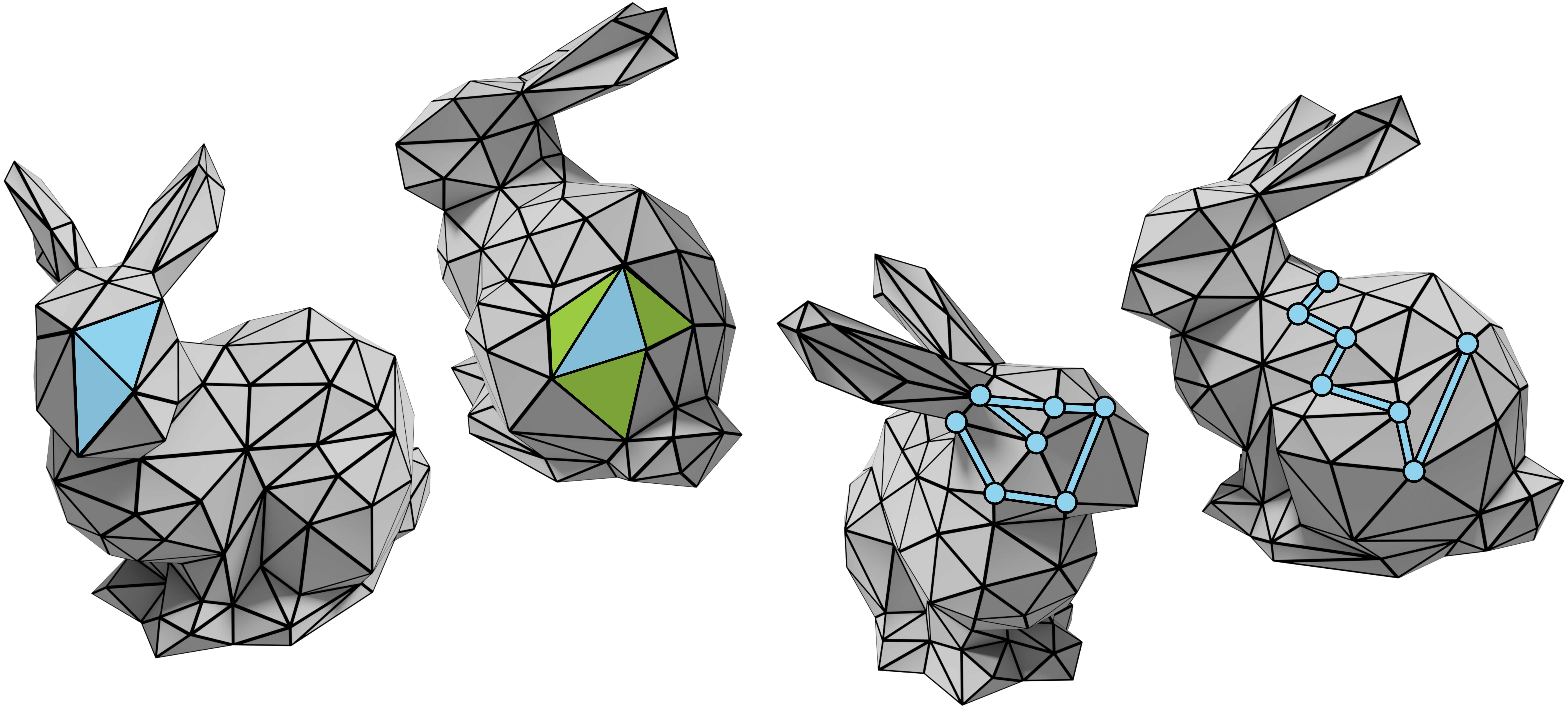
Toric Covers. Maron et. al [SIGGRAPH 2017]

HodgeNet. Smirnov & Solomon [SIGGRAPH 2021]



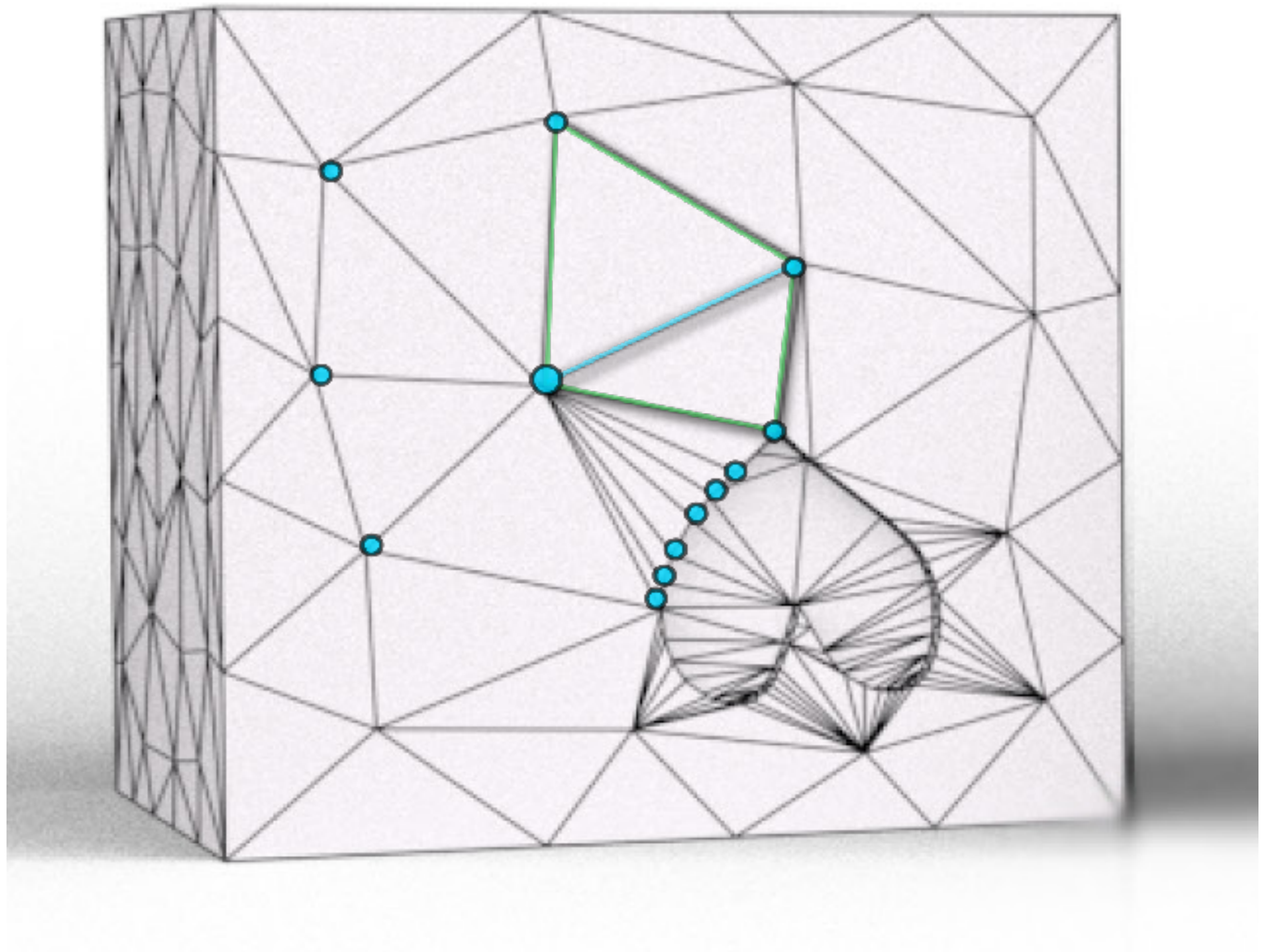
CNNs on Surfaces. Wiersma & Eisemann [SIGGRAPH 2020]

Convolutions on meshes



Fixed size neighborhood

Mesh edges have 4 edge-neighbors



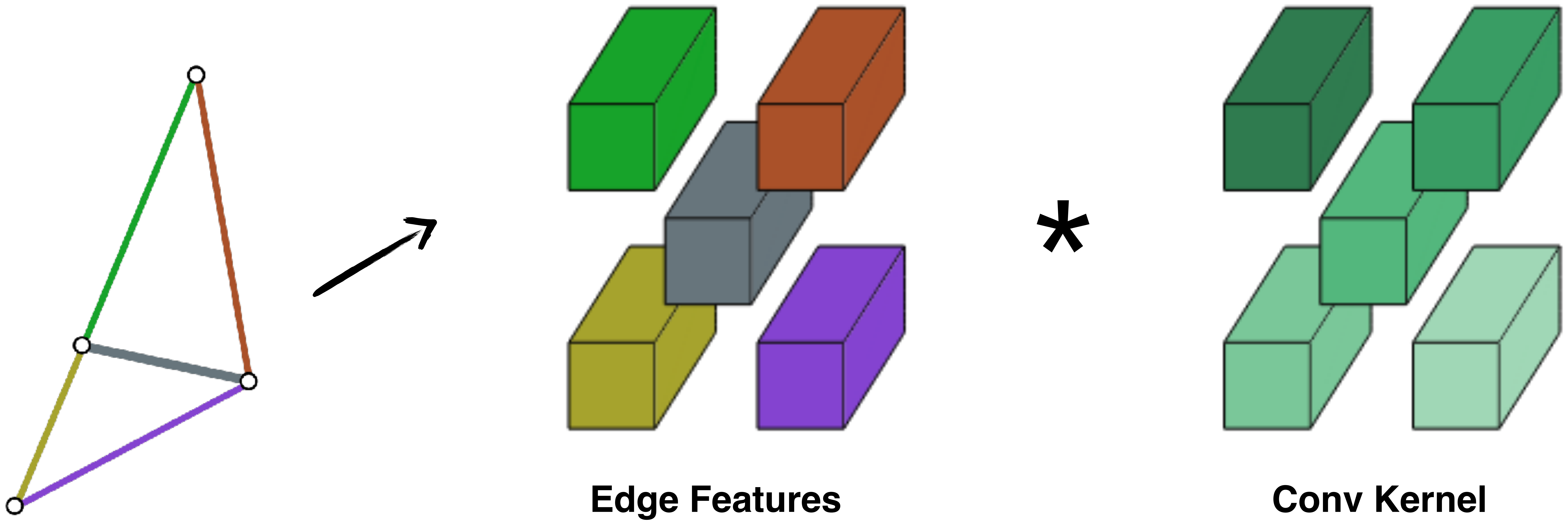
Vertices
<x, y, z>

Edges
<vi, vj>

Faces
<vi, vj, vk>

MeshCNN: a Network with an Edge [SIGGRAPH 2019]

Learn Filters on Edge Features



Edge Features

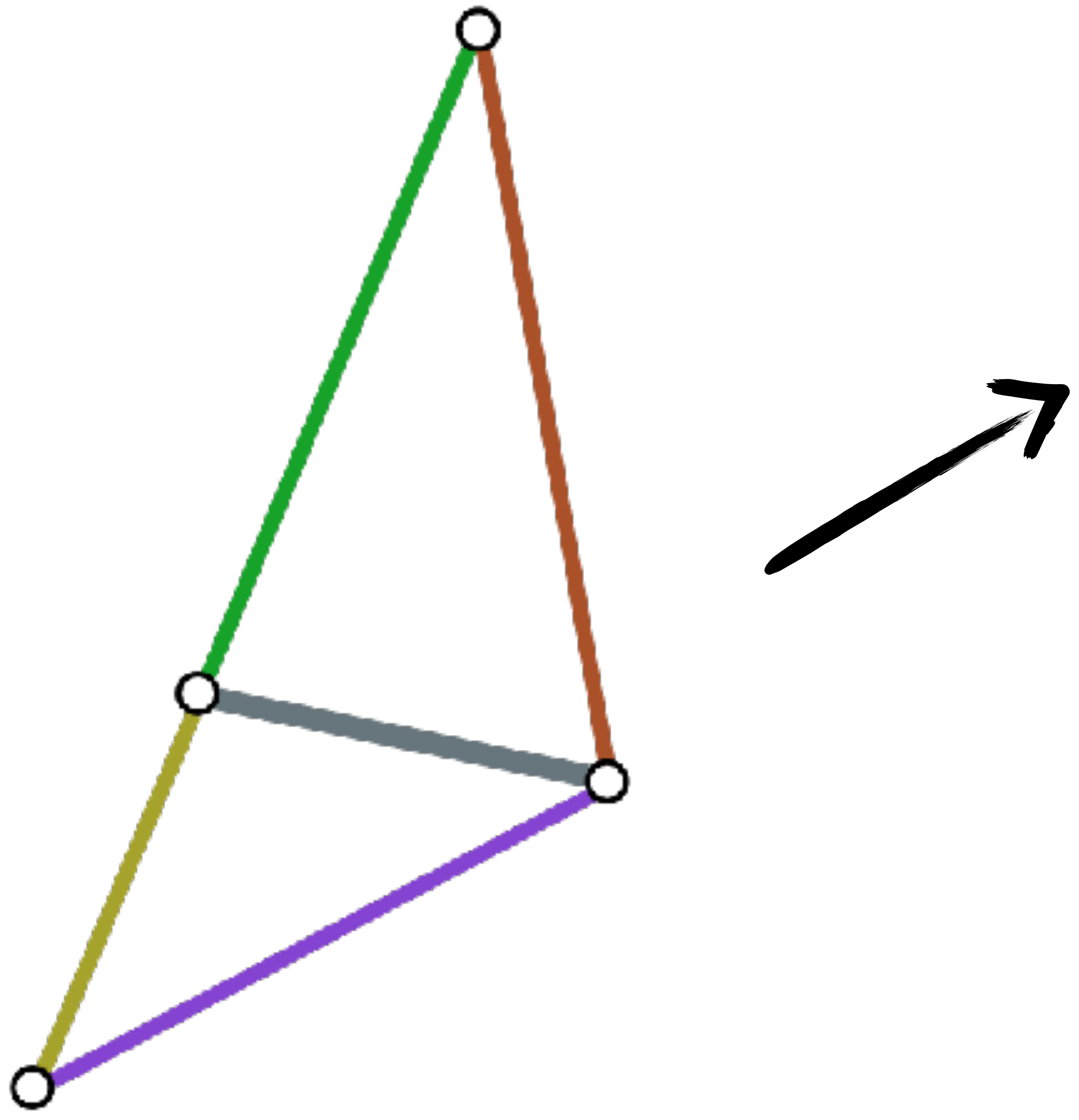
*

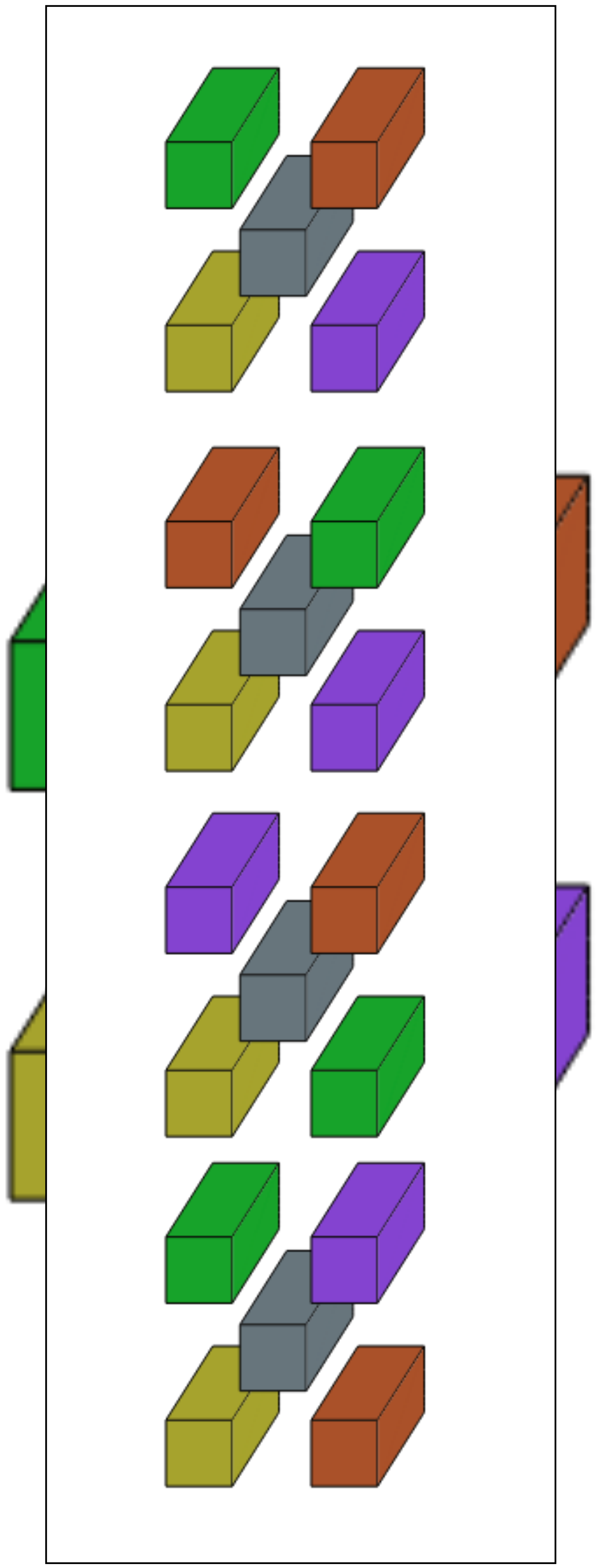
Conv Kernel

MeshCNN: a Network with an Edge [SIGGRAPH 2019]

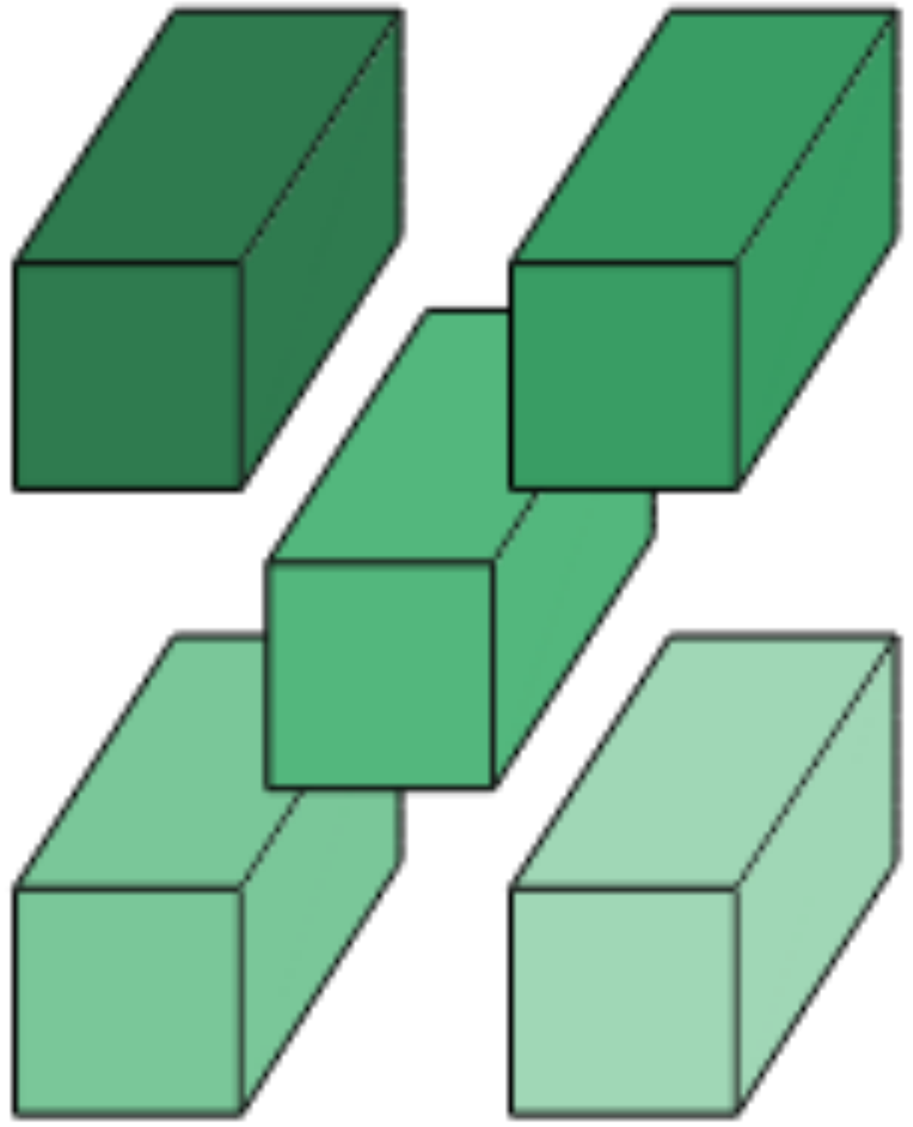
Learn Filters on Edge Features

Different ordering of edge indices





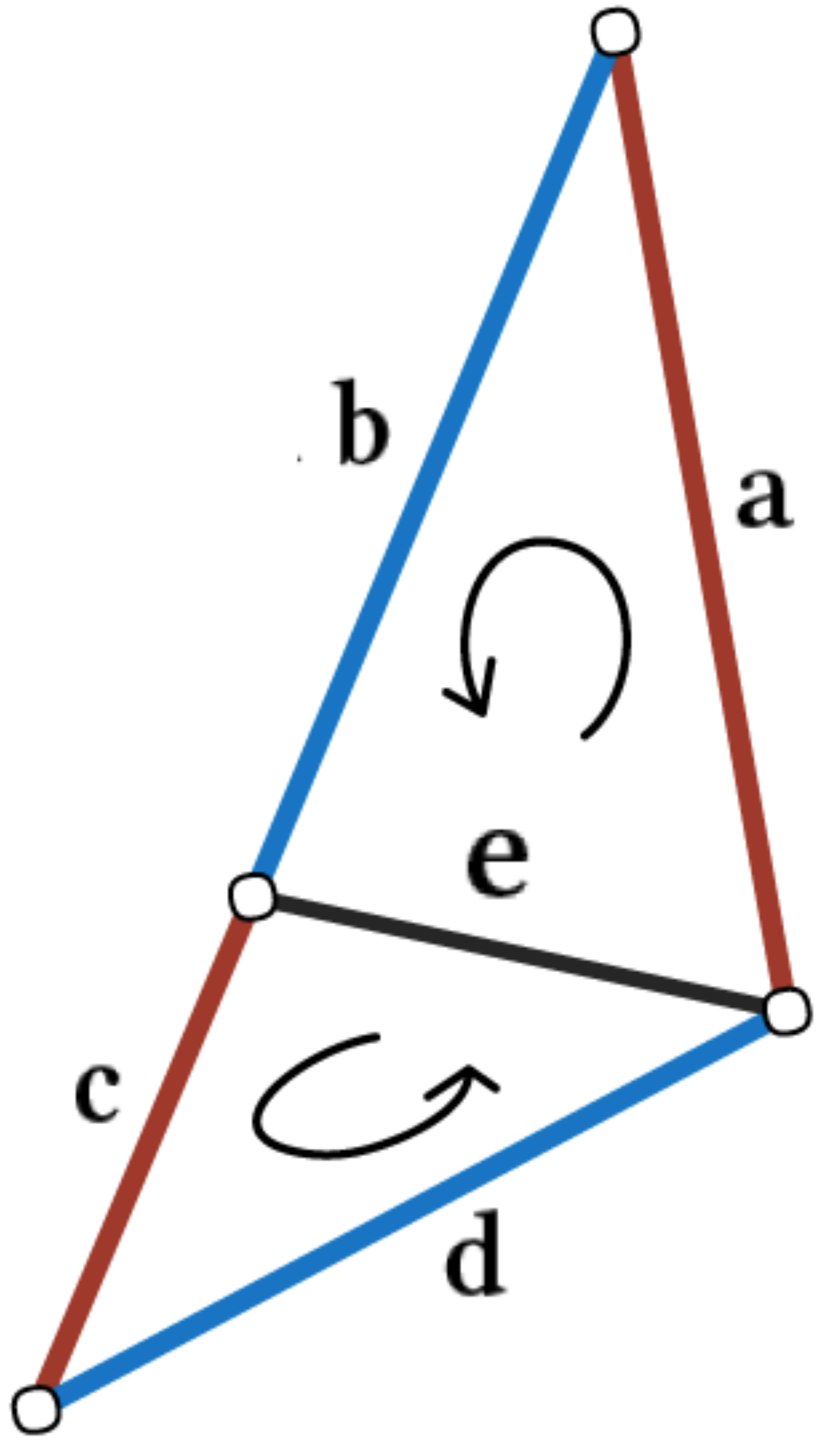
*



Conv Kernel

MeshCNN: a Network with an Edge [SIGGRAPH 2019]

Mesh Convolution Order



Face normal

- Consistent ordering in each face
- Two *valid* orderings

Build symmetric features

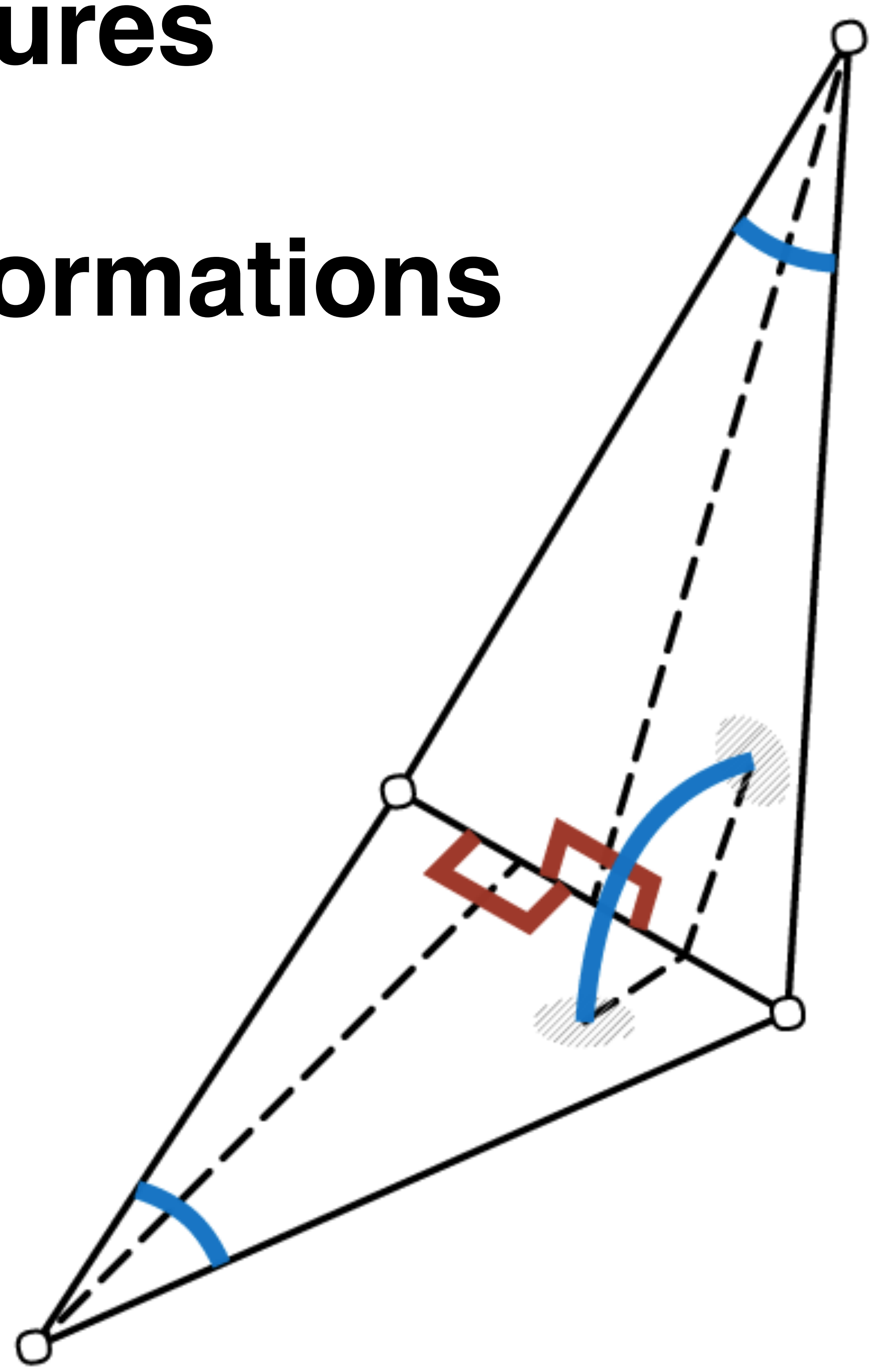
e-> (a+c, la-cl, b+d, lb-dl)

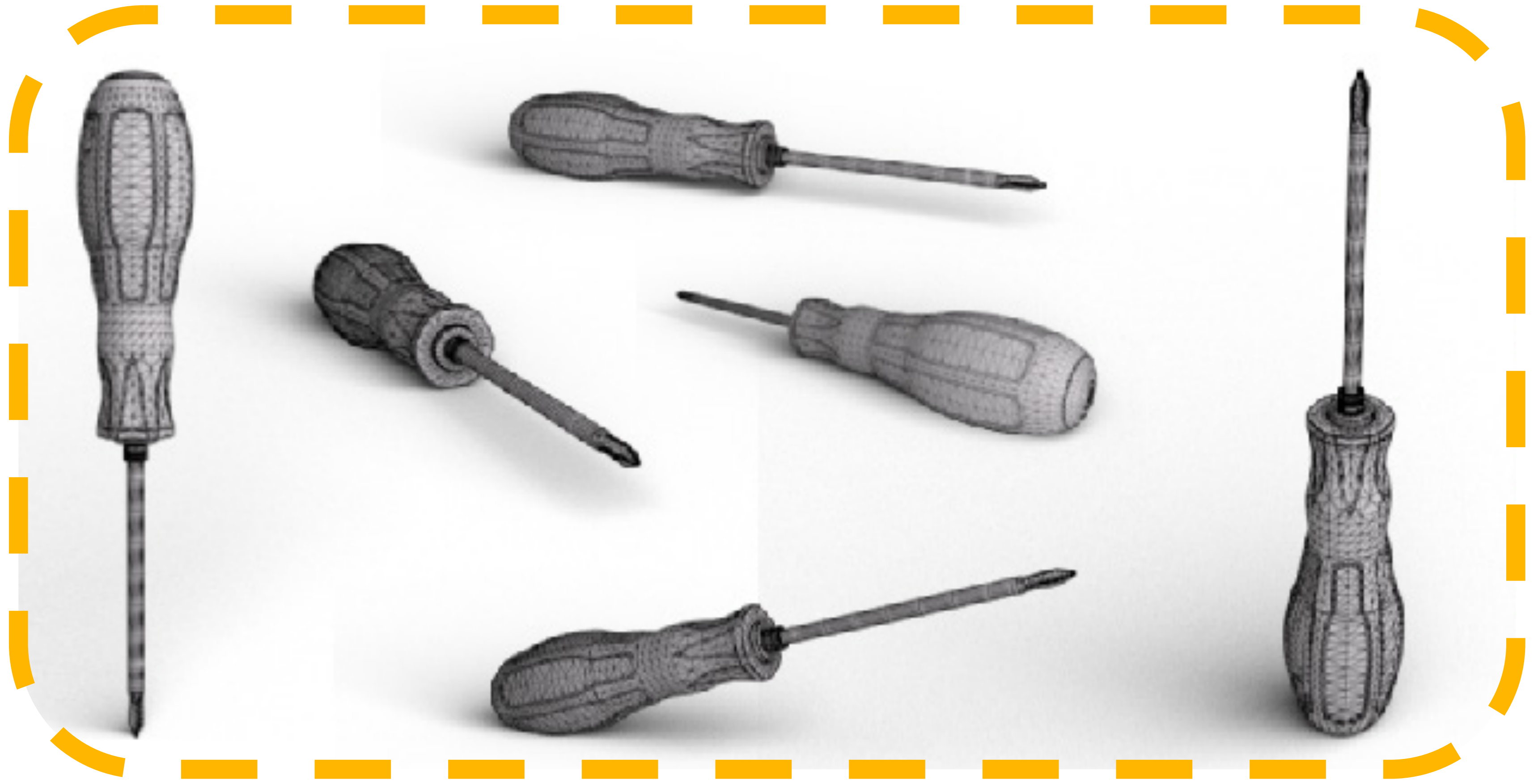
MeshCNN: a Network with an Edge [SIGGRAPH 2019]

Input edge features

Relative geometric features

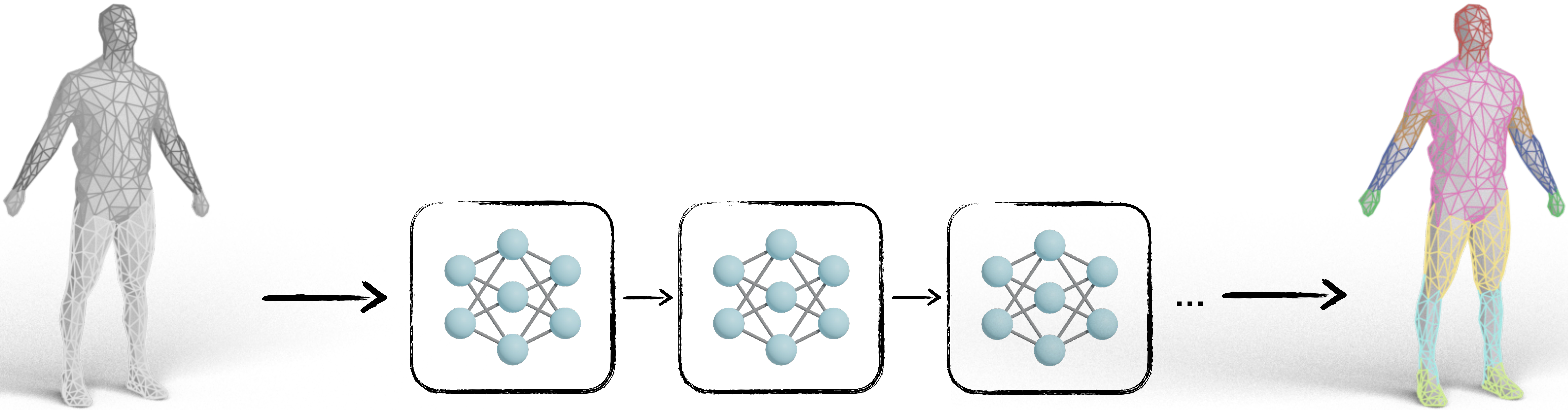
Invariant to rigid transformations





MeshCNN: a Network with an Edge [SIGGRAPH 2019]

Recap: learning local descriptors



Input mesh

Convolution

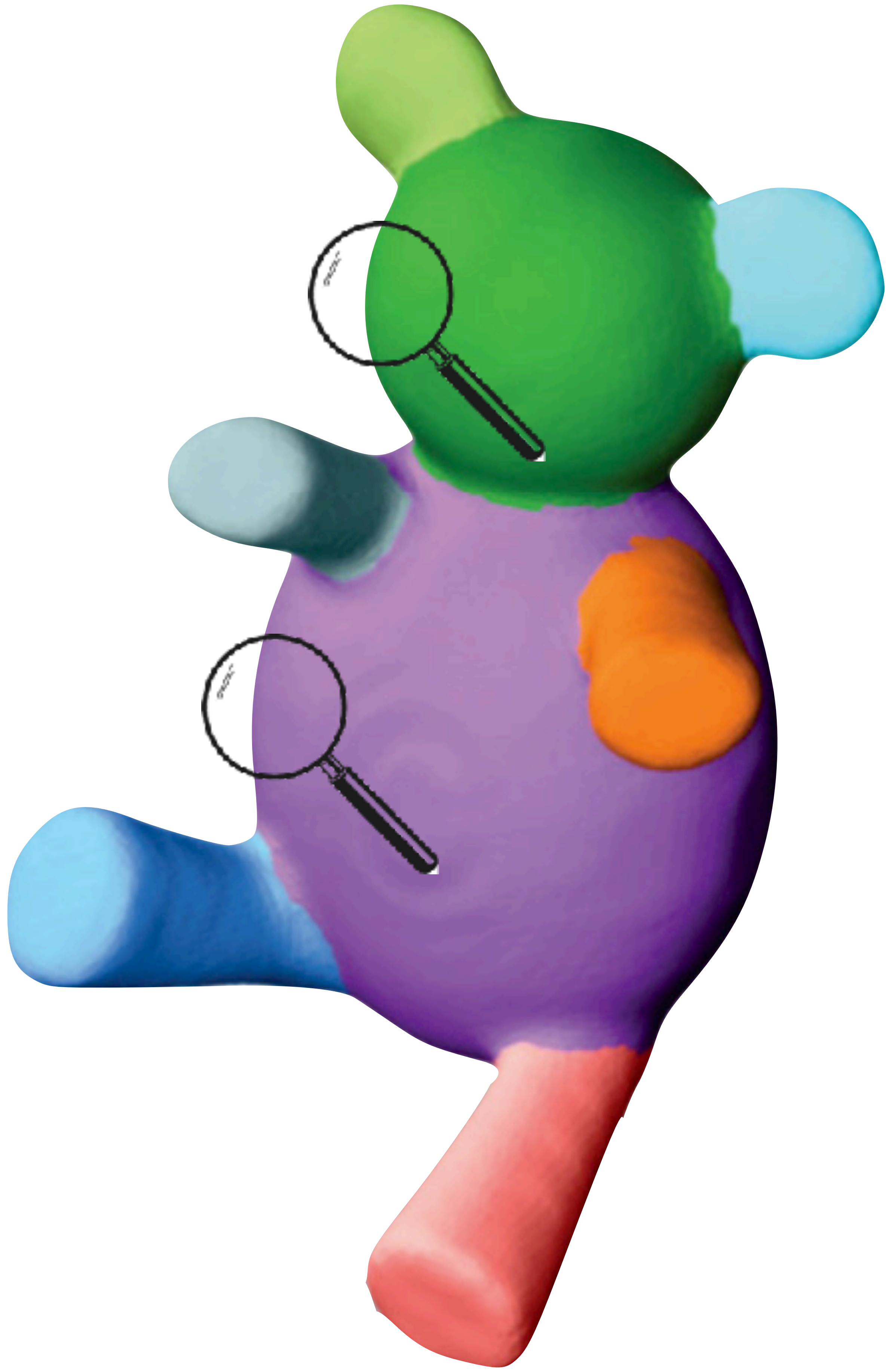
Convolution

Convolution

...

Per edge attribute

Incorporating more context



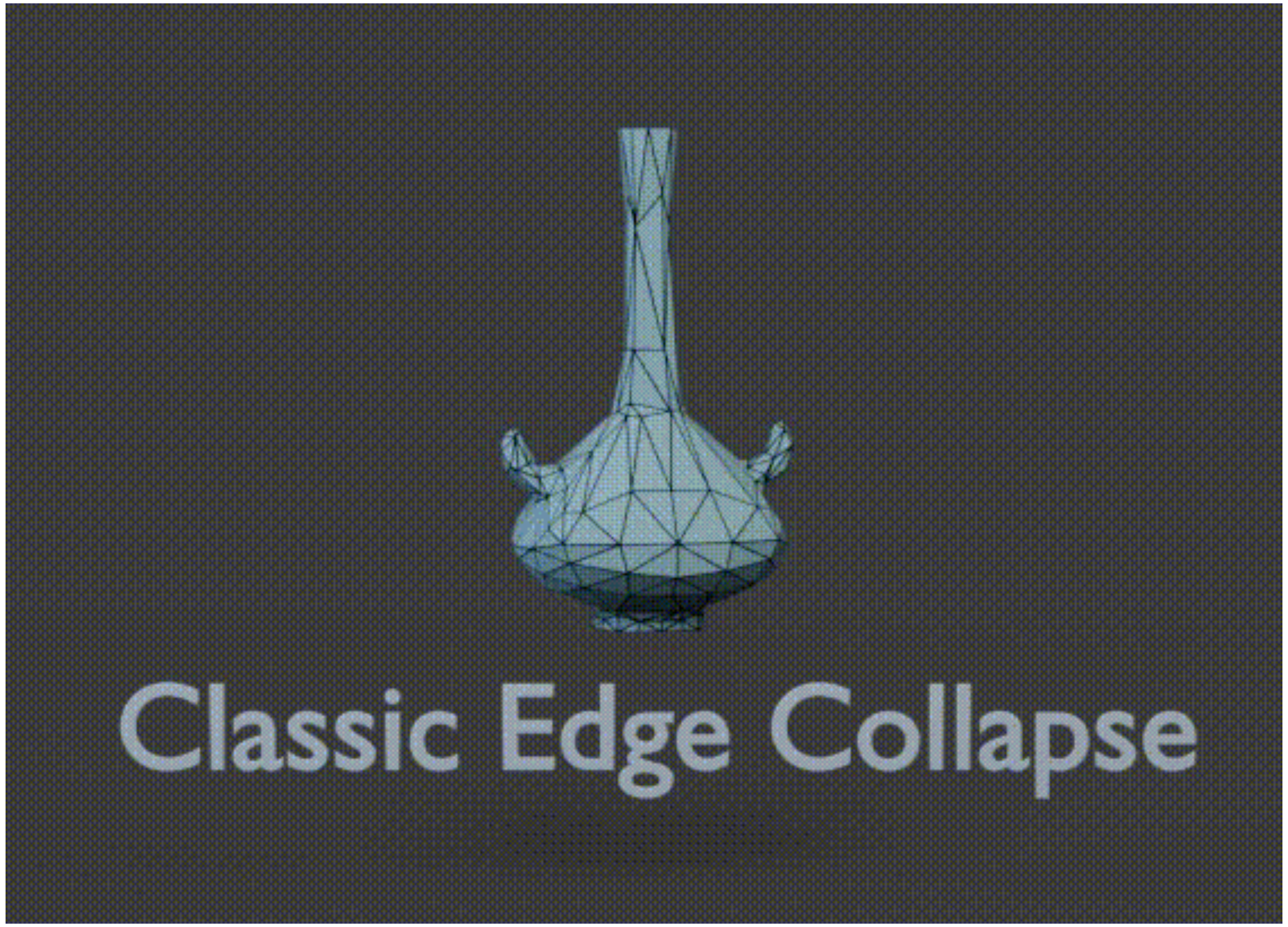
Inspiration: image pooling

4	6	1	1
1	3	1	3
4	0	0	8
8	5	4	0

Input (4x4)

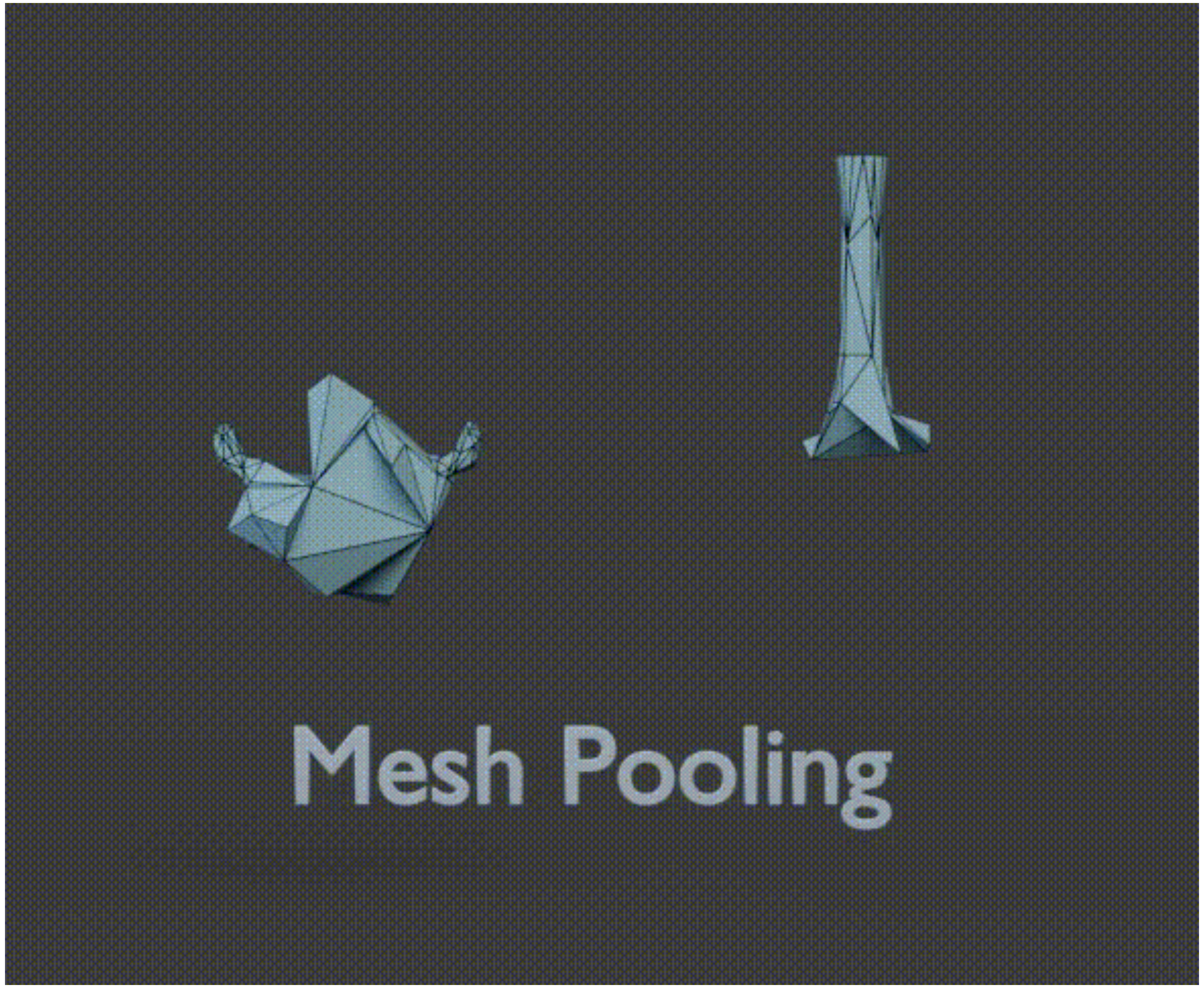
Output (2x2)

Mesh pooling via edge collapse



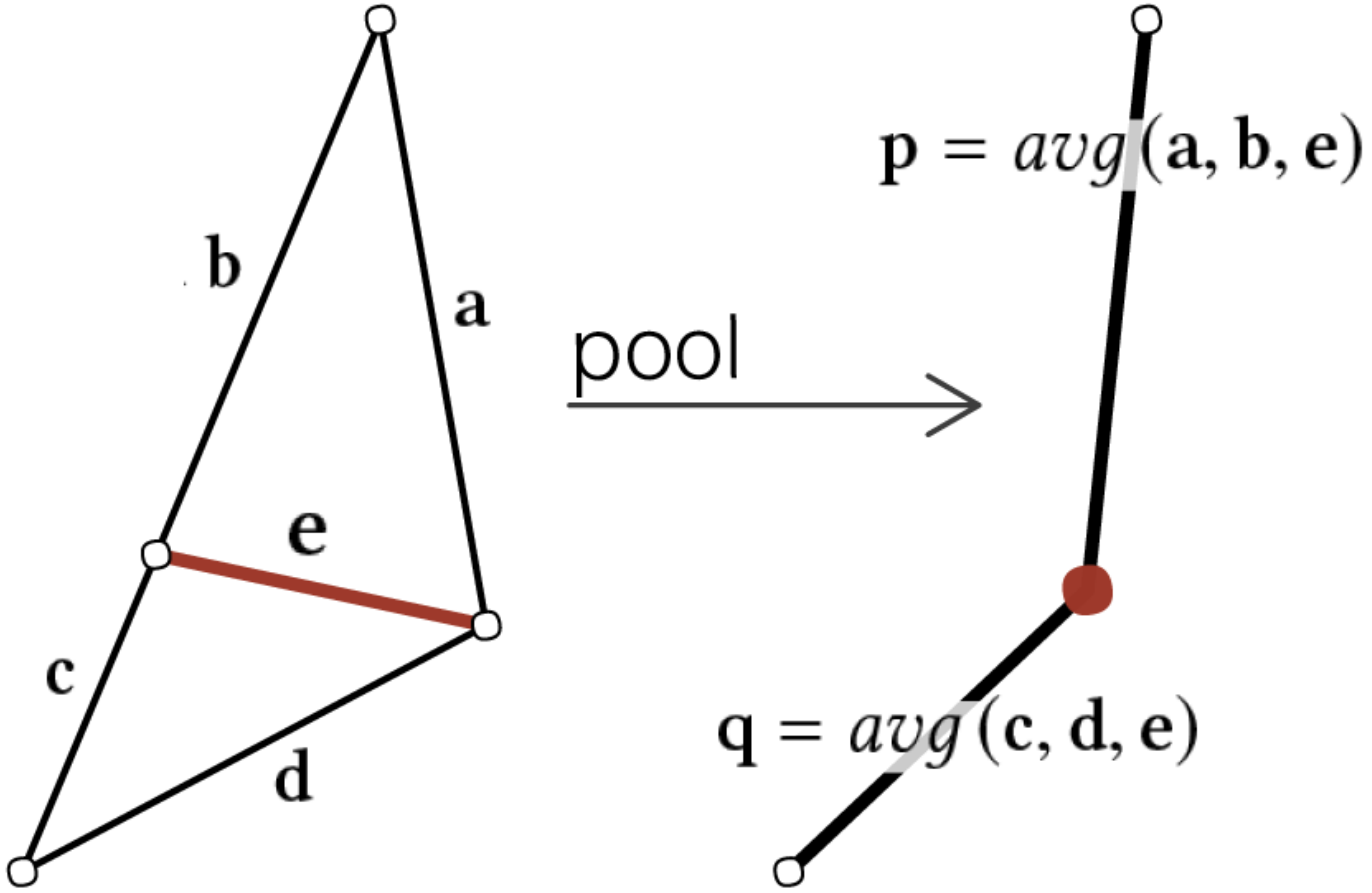
MeshCNN: a Network with an Edge [SIGGRAPH 2019]

Mesh pooling via edge collapse



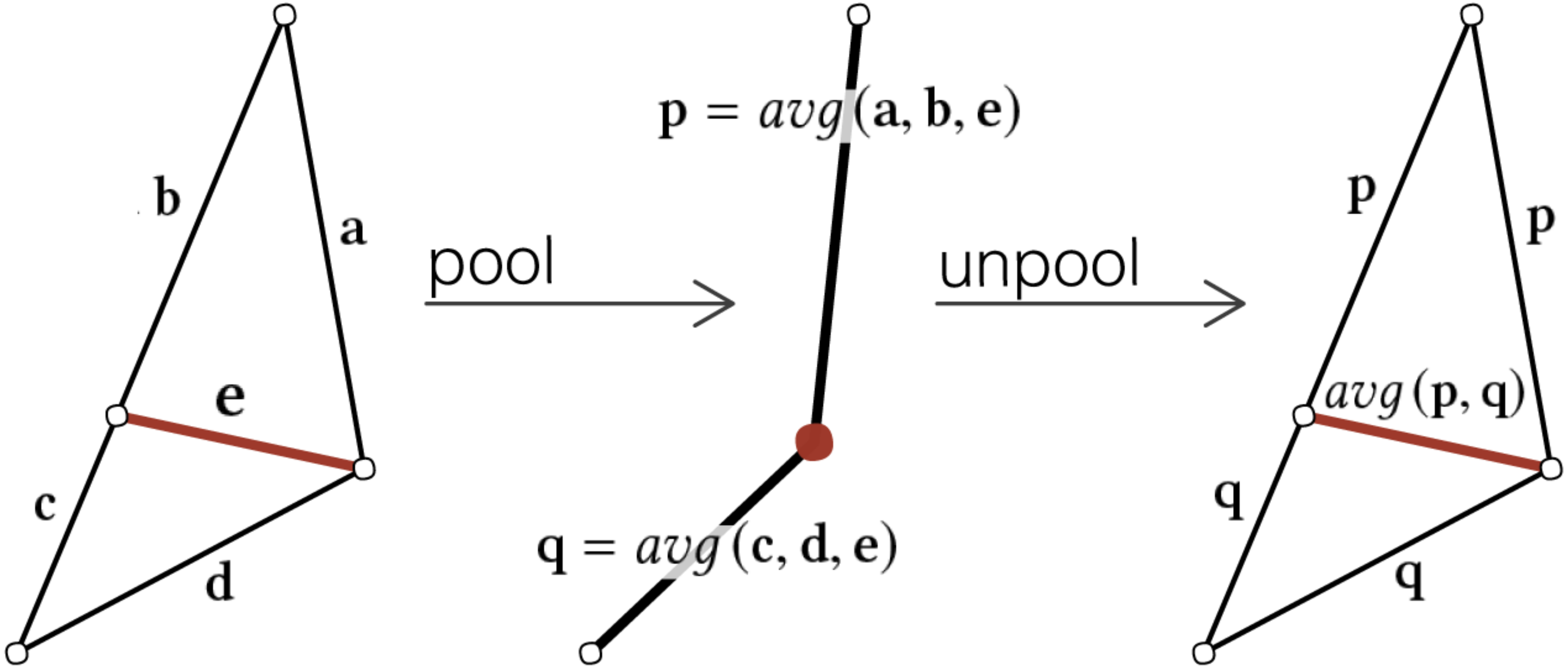
MeshCNN: a Network with an Edge [SIGGRAPH 2019]

Mesh pooling via edge collapse



MeshCNN: a Network with an Edge [SIGGRAPH 2019]

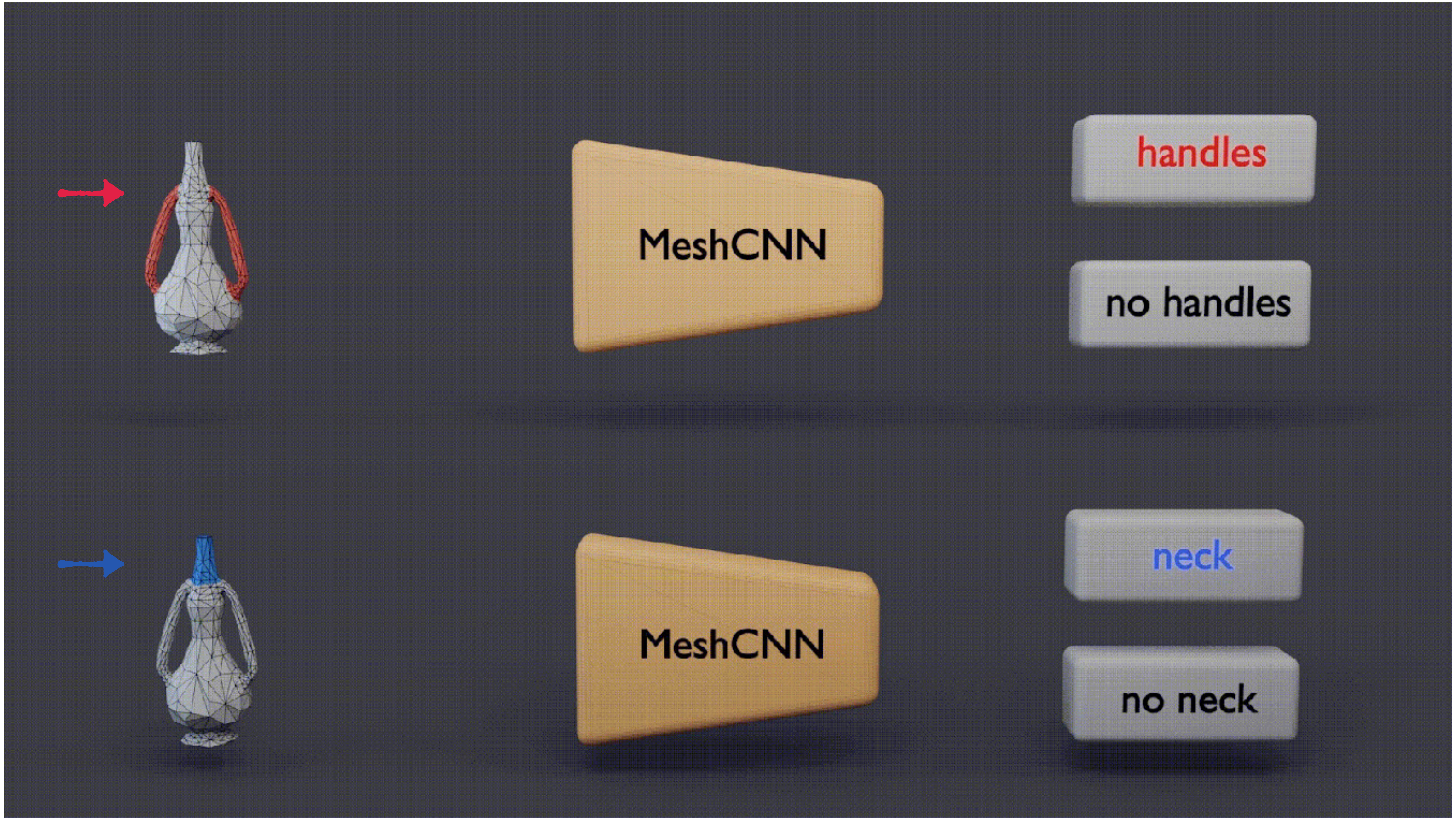
Mesh unpooling


b
a
e
c
d
pool
$\mathbf{p} = avg(\mathbf{a}, \mathbf{b}, \mathbf{e})$
$\mathbf{q} = avg(\mathbf{c}, \mathbf{d}, \mathbf{e})$
unpool
p
p
$avg(\mathbf{p}, \mathbf{q})$
q
q

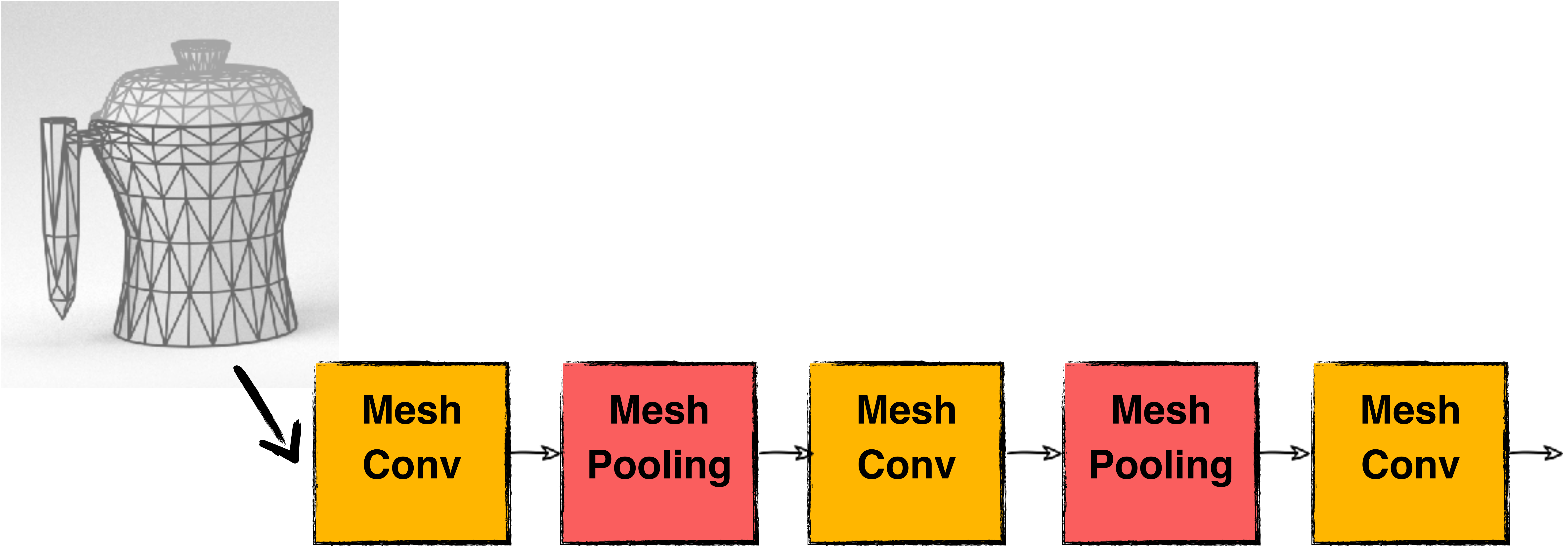
MeshCNN: a Network with an Edge [SIGGRAPH 2019]

Different tasks

Different simplifications



MeshCNN



Mesh Conv

Mesh Pooling

Mesh Conv

Mesh Pooling

Mesh Conv

MeshCNN Segmentation Overview

Mesh Conv

Mesh Pooling

Mesh Conv

Mesh Pooling

Mesh Conv

Mesh Unpooling

Mesh Conv

Mesh Unpooling

MeshCNN Classification Overview

Mesh Conv

Mesh Pooling

Mesh Conv

Mesh Pooling

Mesh Conv

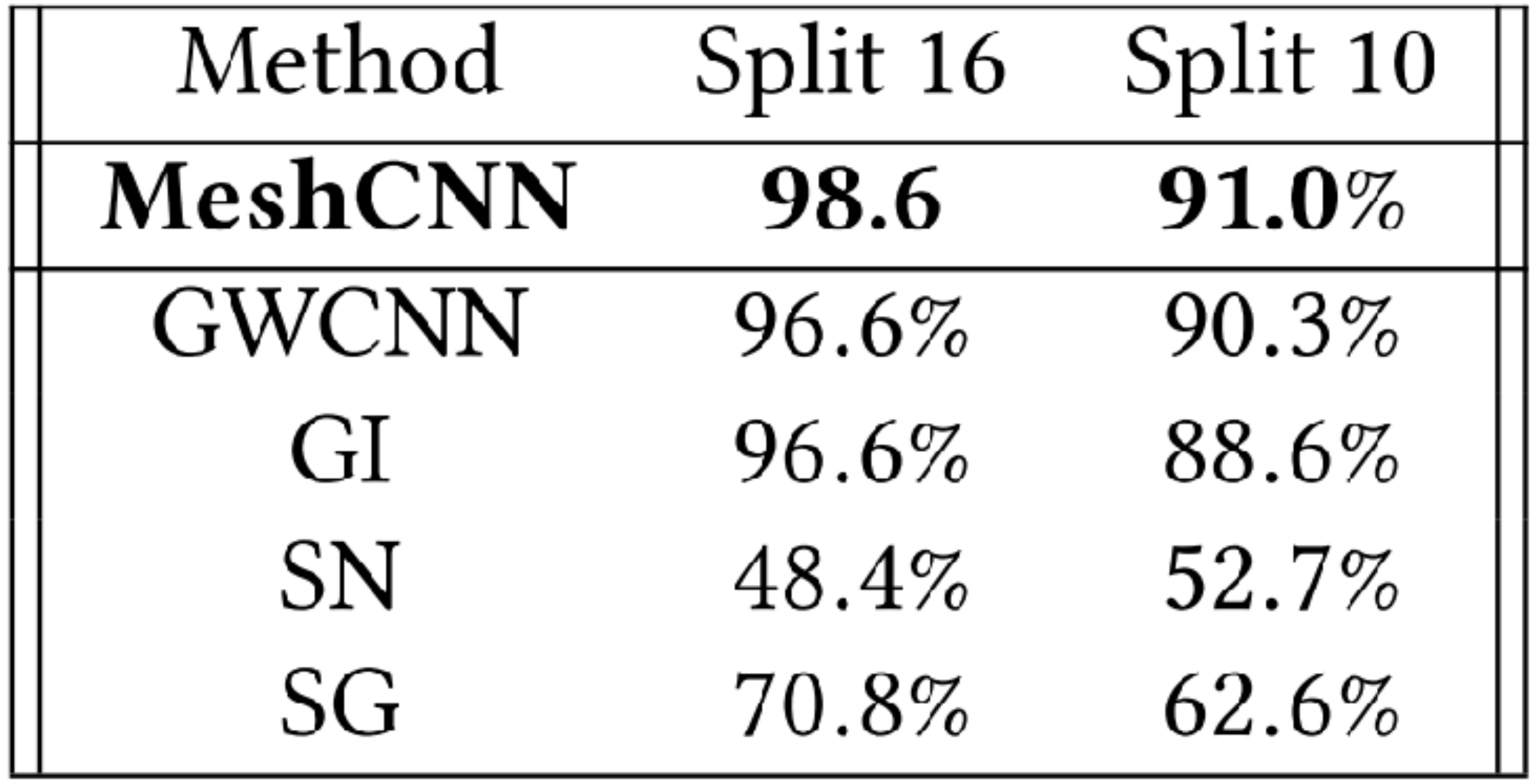
Mesh Unpooling

Fully Connected

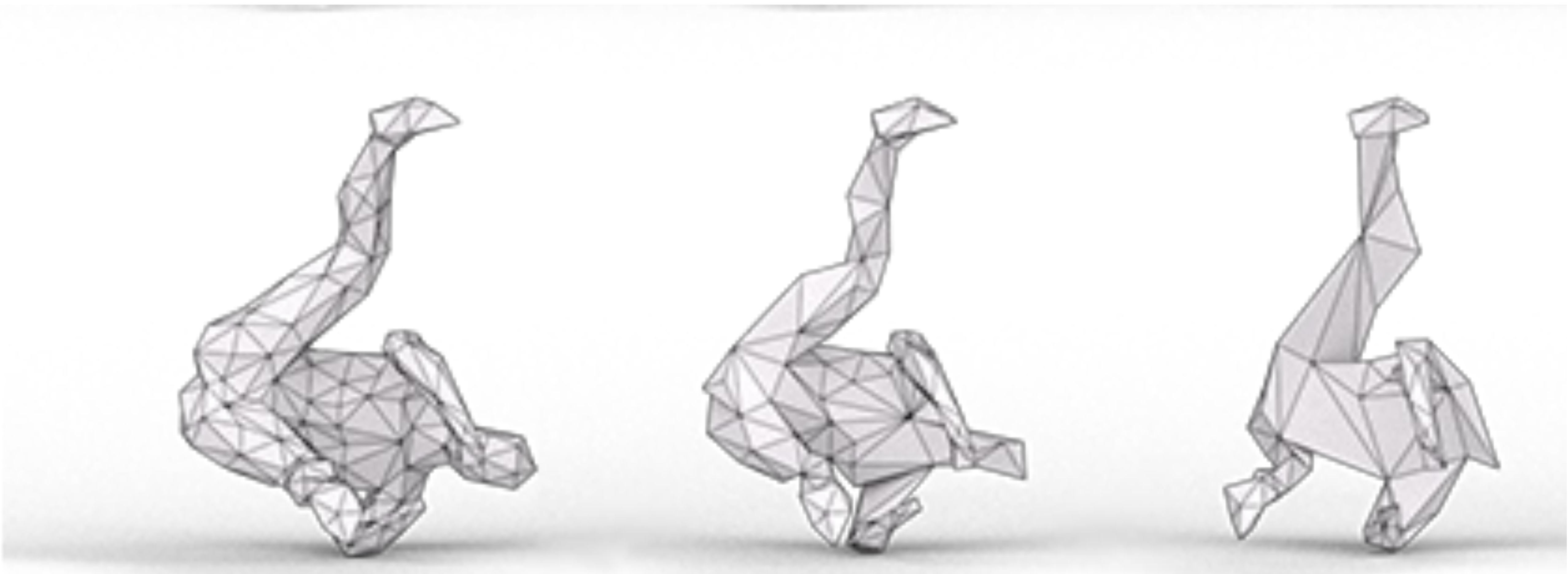
Fully Connected

Human

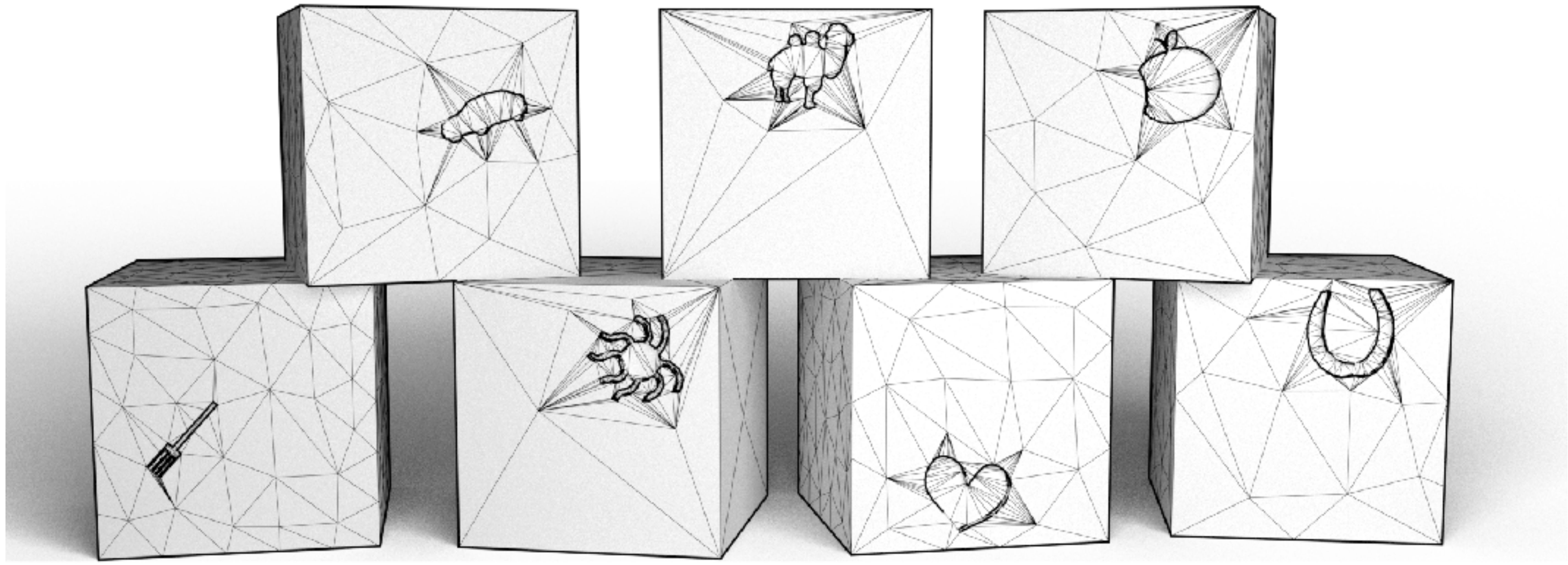
Shape Classification



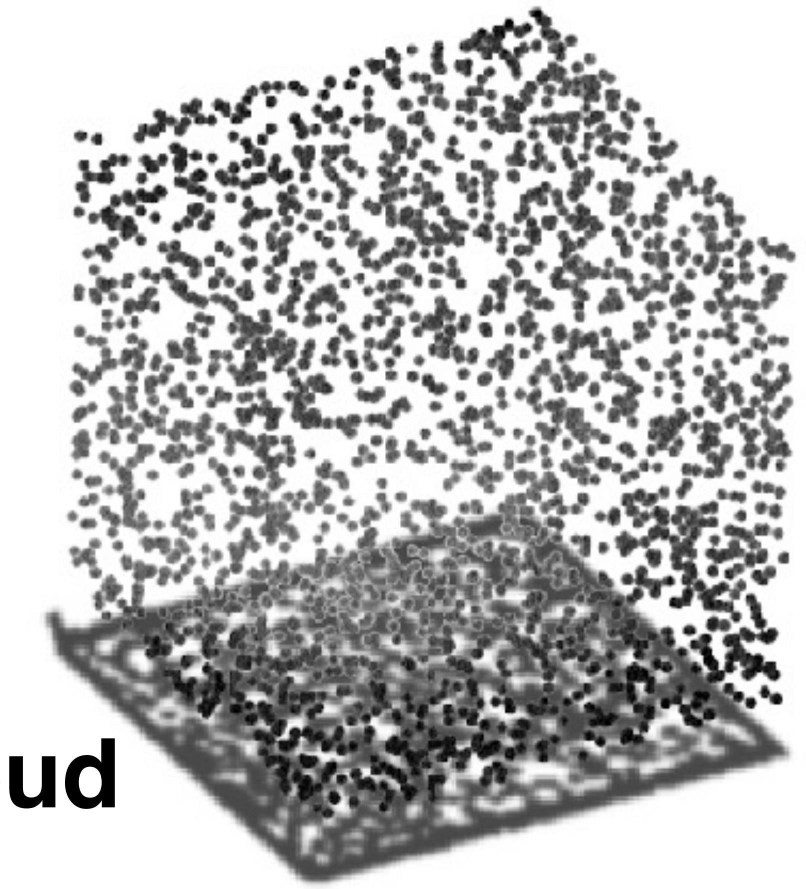
Method	Split 16	Split 10
MeshCNN	**98.6**	**91.0**%
GWCNN	96.6%	90.3%
GI	96.6%	88.6%
SN	48.4%	52.7%
SG	70.8%	62.6%



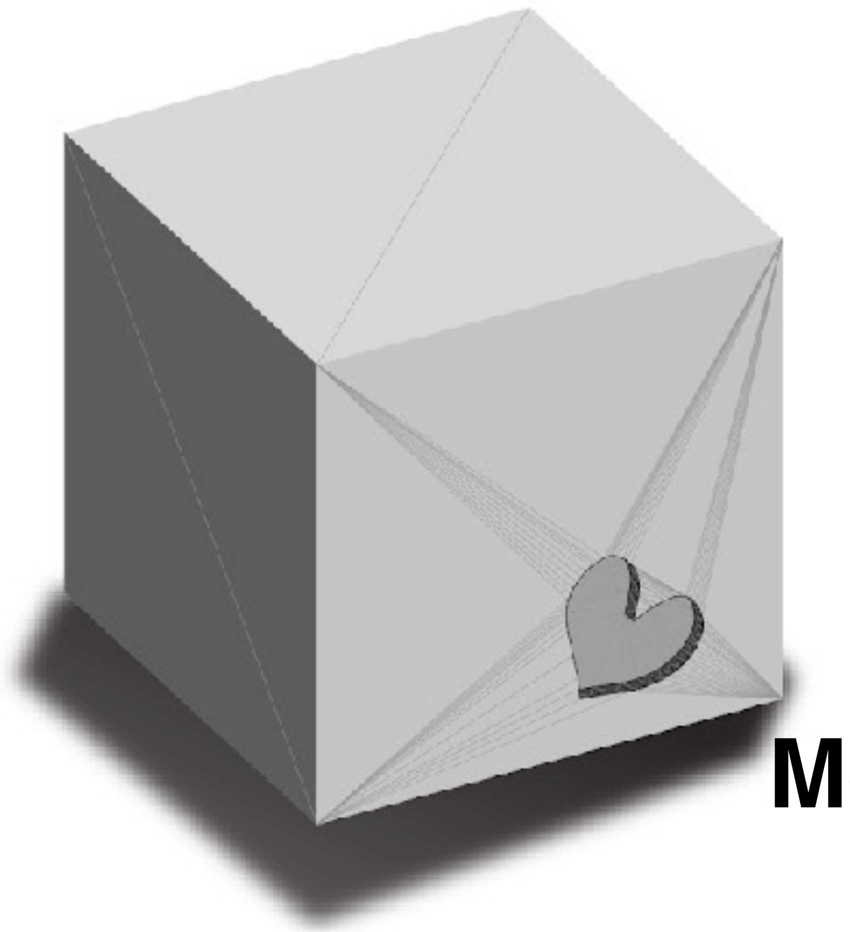
Cube Engraving Classification



Which engraving does this have?

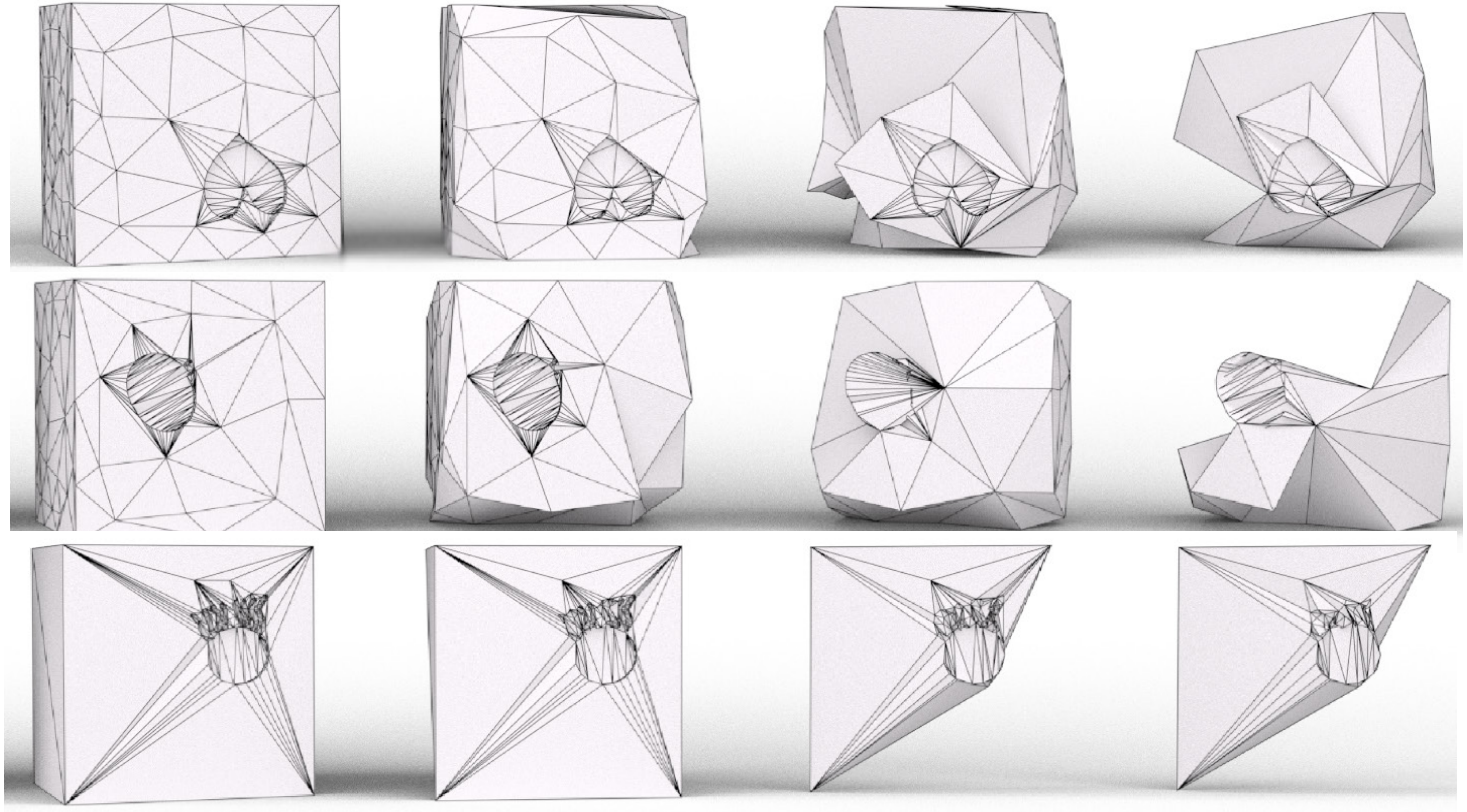


64.26% Point cloud



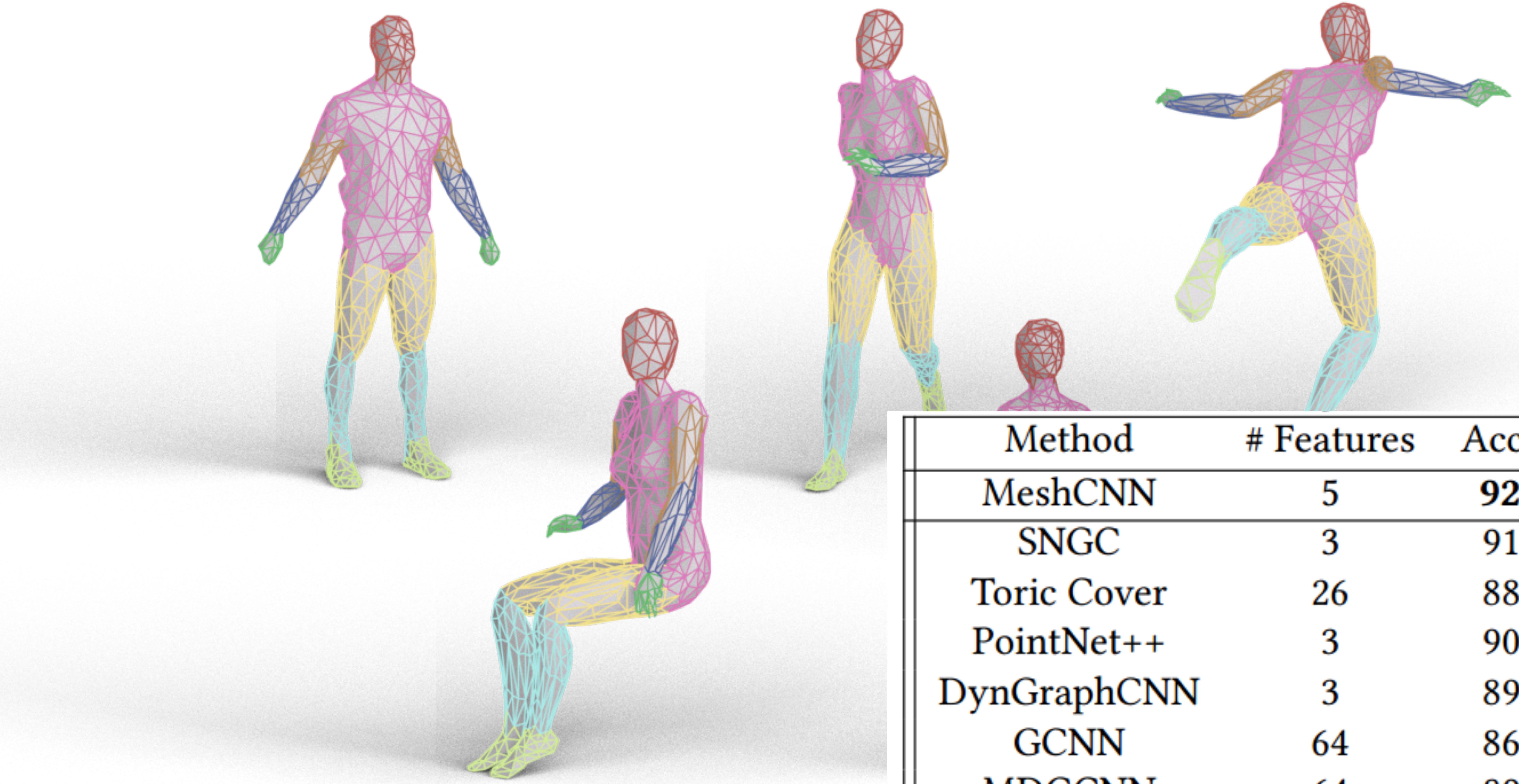
Mesh 92.16%

Intermediate Mesh Pooling



MeshCNN: a Network with an Edge [SIGGRAPH 2019]

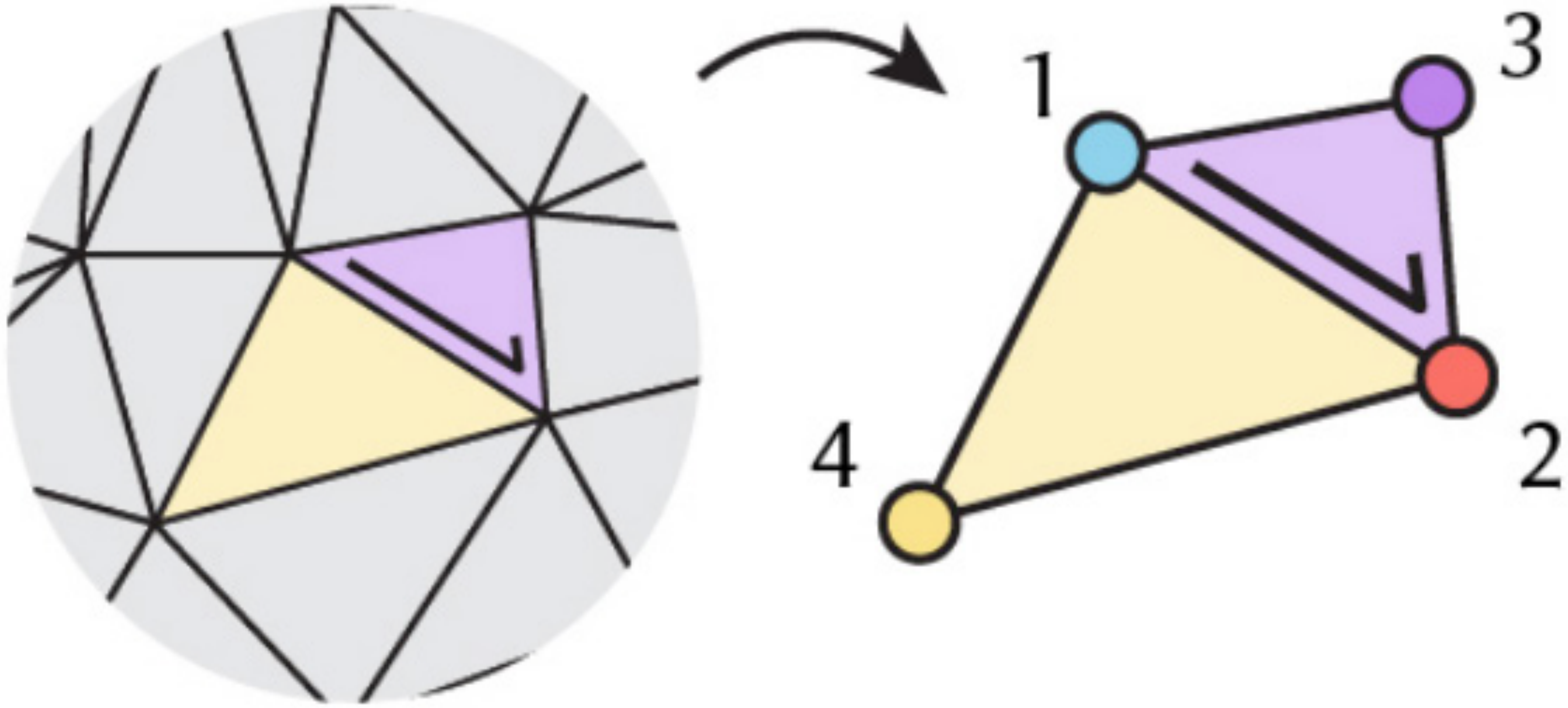
Human Segmentation



Method	# Features	Accuracy
MeshCNN	5	**92.30%**
SNGC	3	91.02%
Toric Cover	26	88.00%
PointNet++	3	90.77%
DynGraphCNN	3	89.72%
GCNN	64	86.40%
MDGCNN	64	89.47%

MeshCNN: a Network with an Edge [SIGGRAPH 2019]

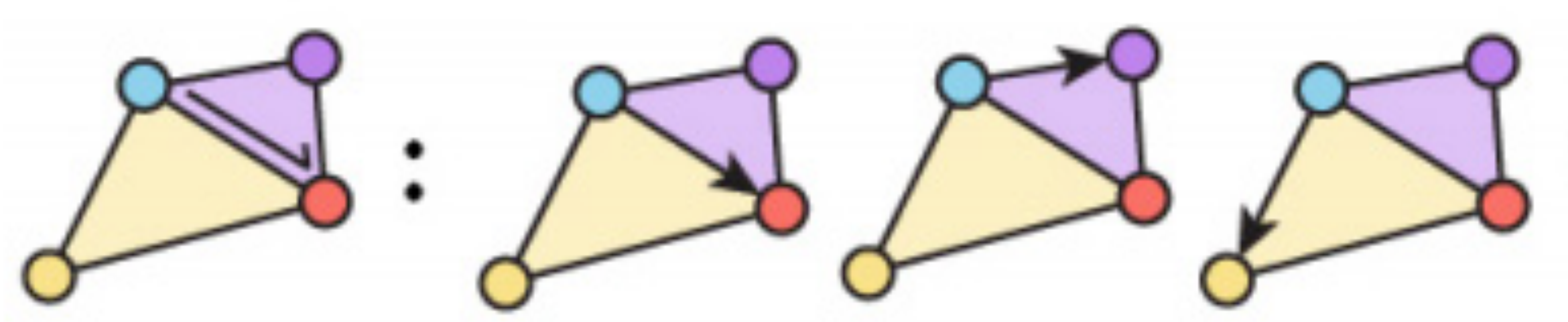
Convolutions on half-edges



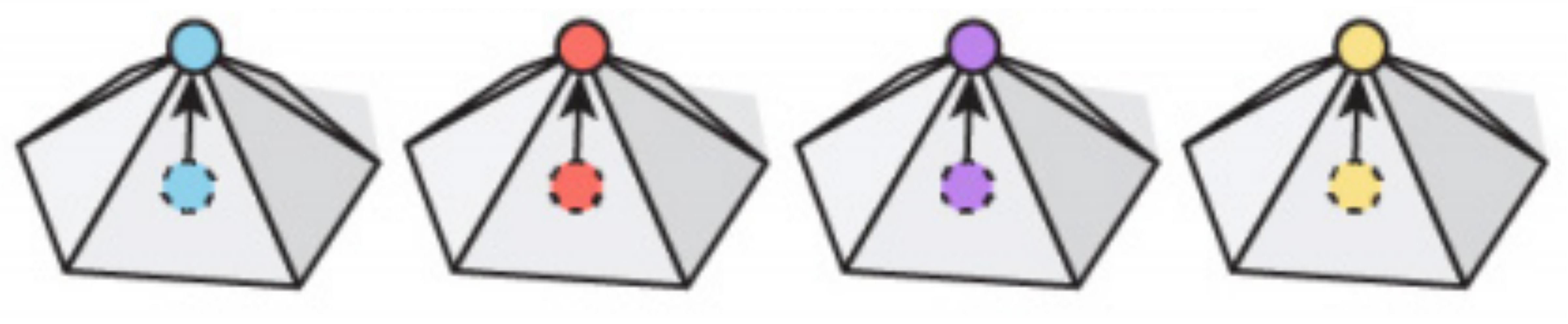
Neural Subdivision. Liu et al [SIGGRAPH 2020]

Half-edge input features

Edge vectors



Differential coordinates

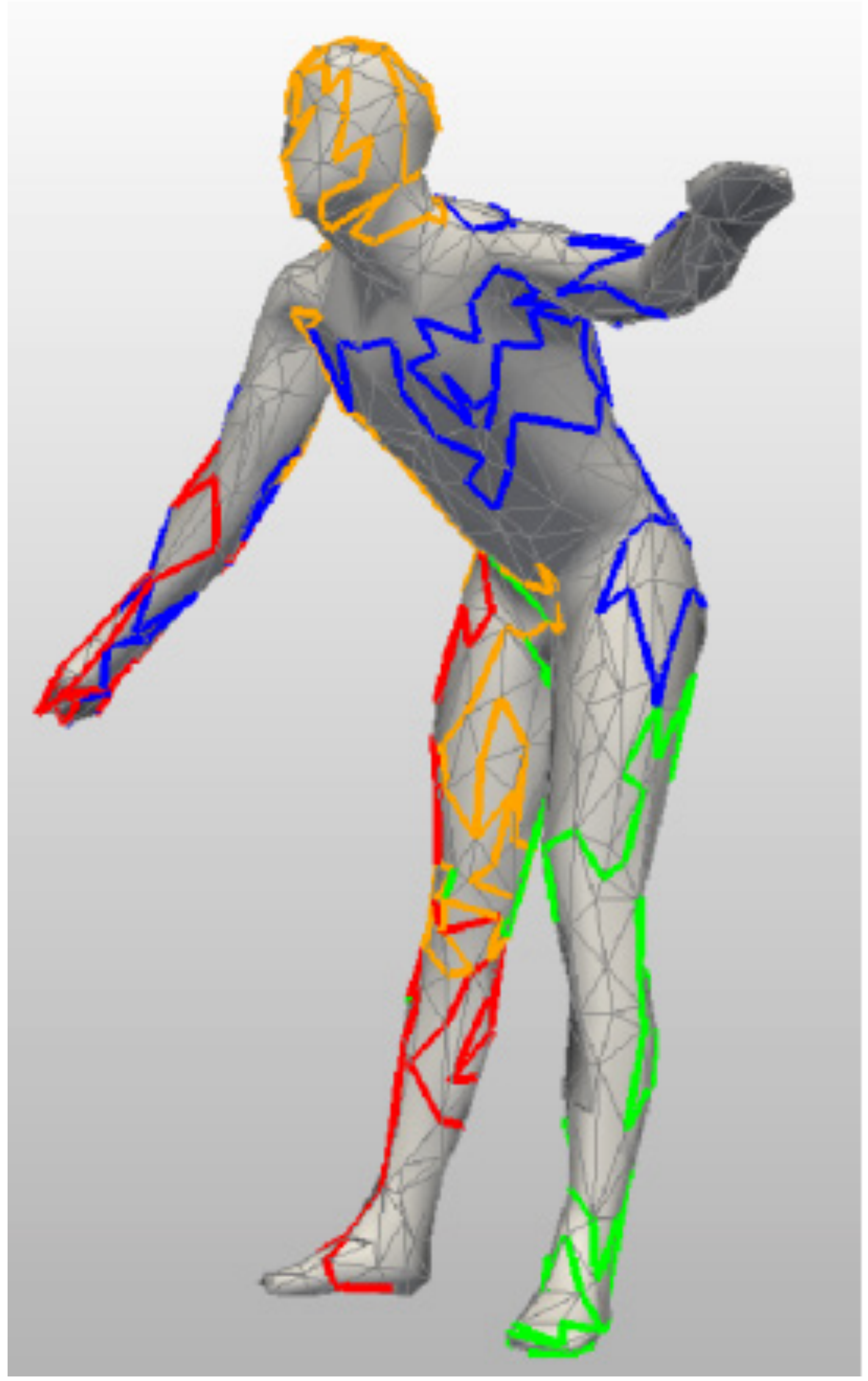


Neural Subdivision. Liu et al [SIGGRAPH 2020]

Input mesh coarsening

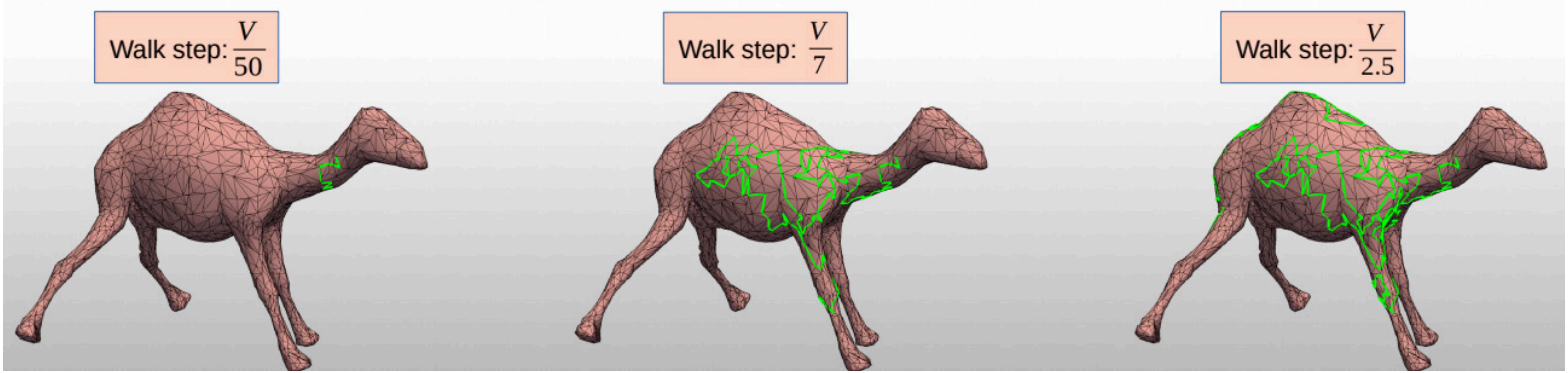
Neural Subdivision. Liu et al [SIGGRAPH 2020]

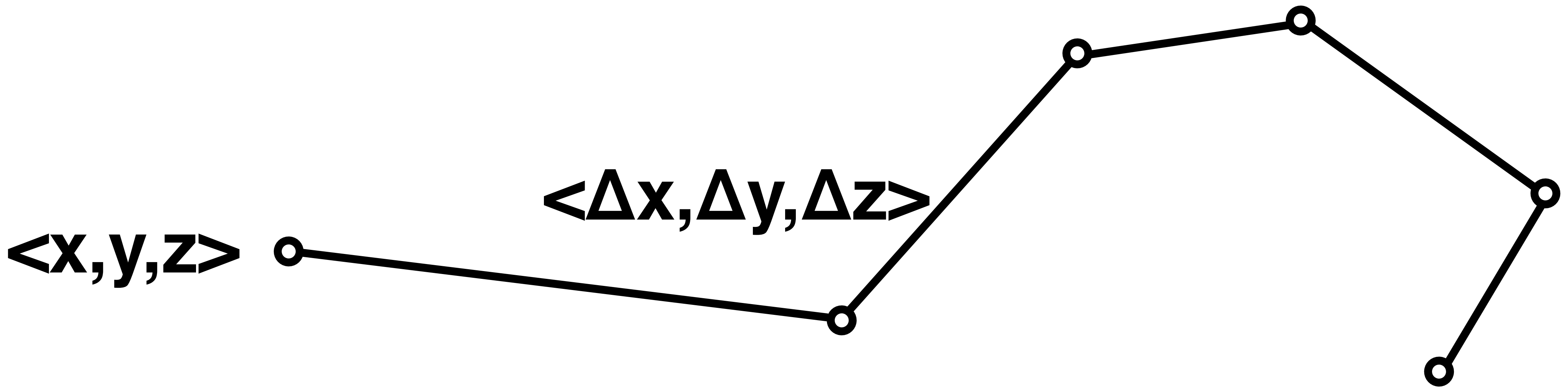
Learning on random walks



MeshWalker. Lerner & Tal [SIGGRAPH Asia 2020]

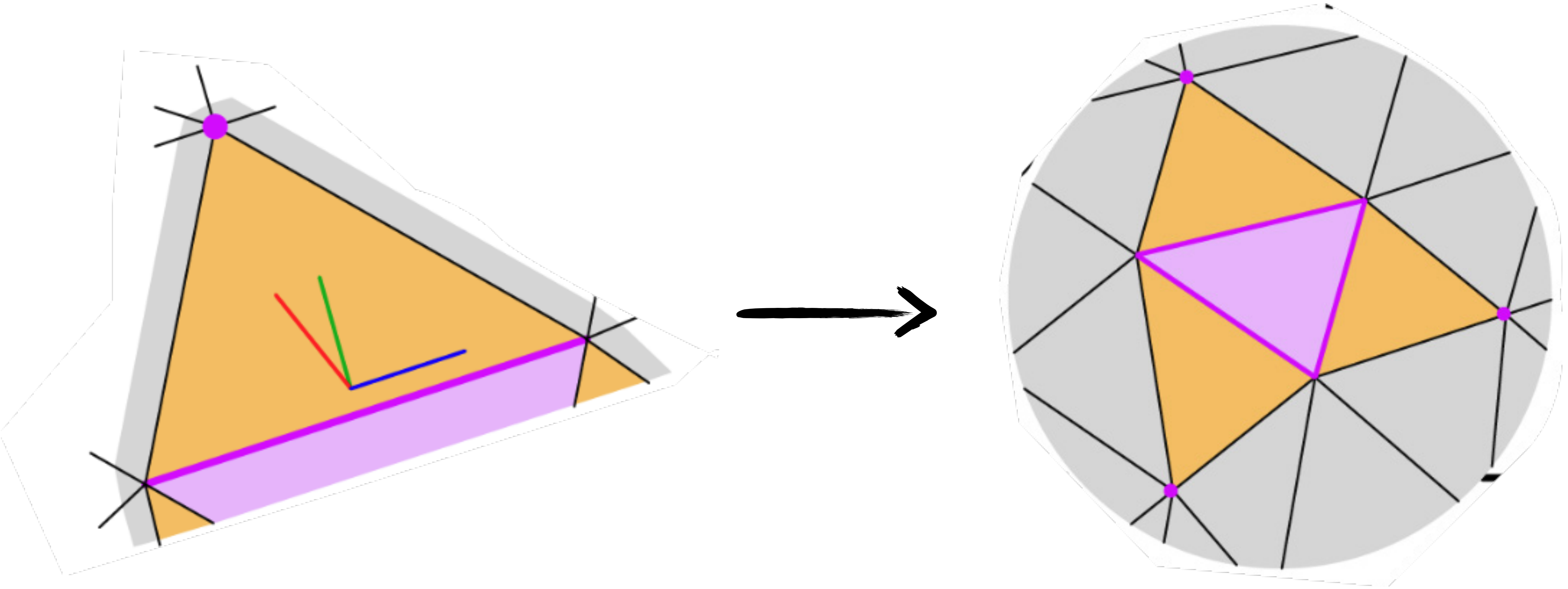
Learning on random walks





MeshWalker. Lerner & Tal [SIGGRAPH Asia 2020]

Convolutions on mesh faces

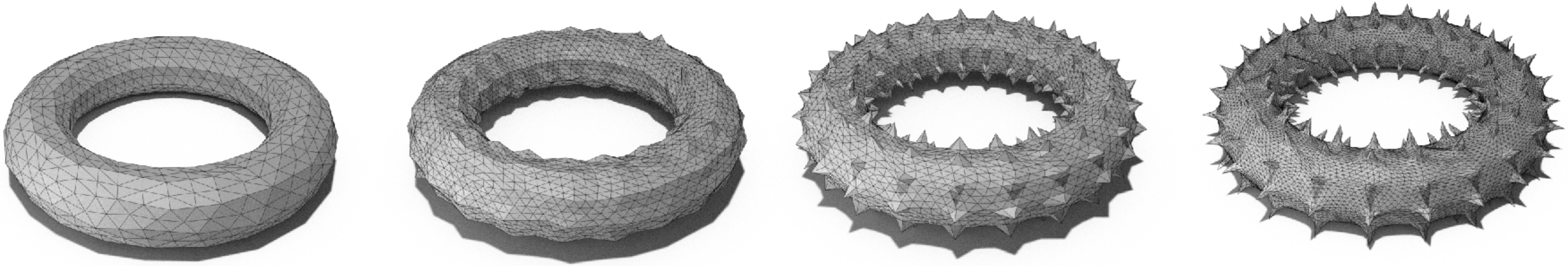


Input features

Face-based convolution

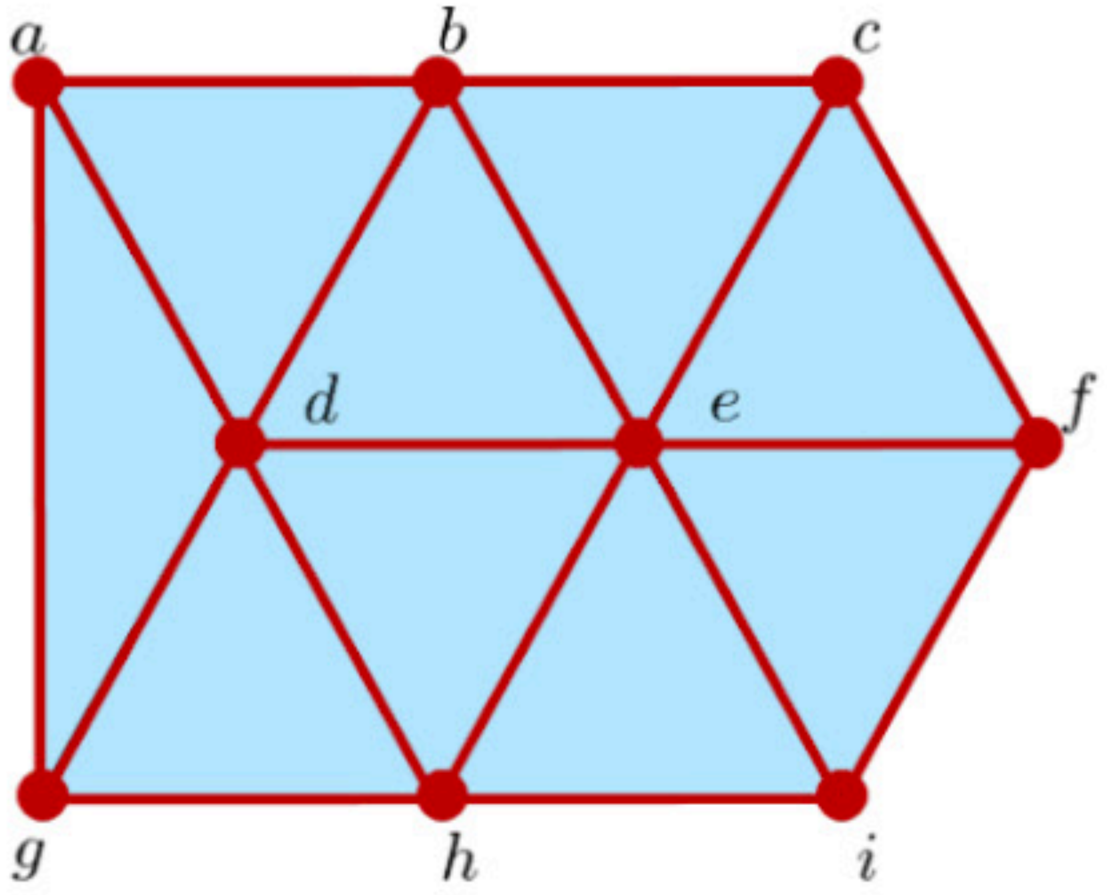
Deep Geometric Texture Synthesis [SIGGRAPH 2020]

Input mesh untexturing

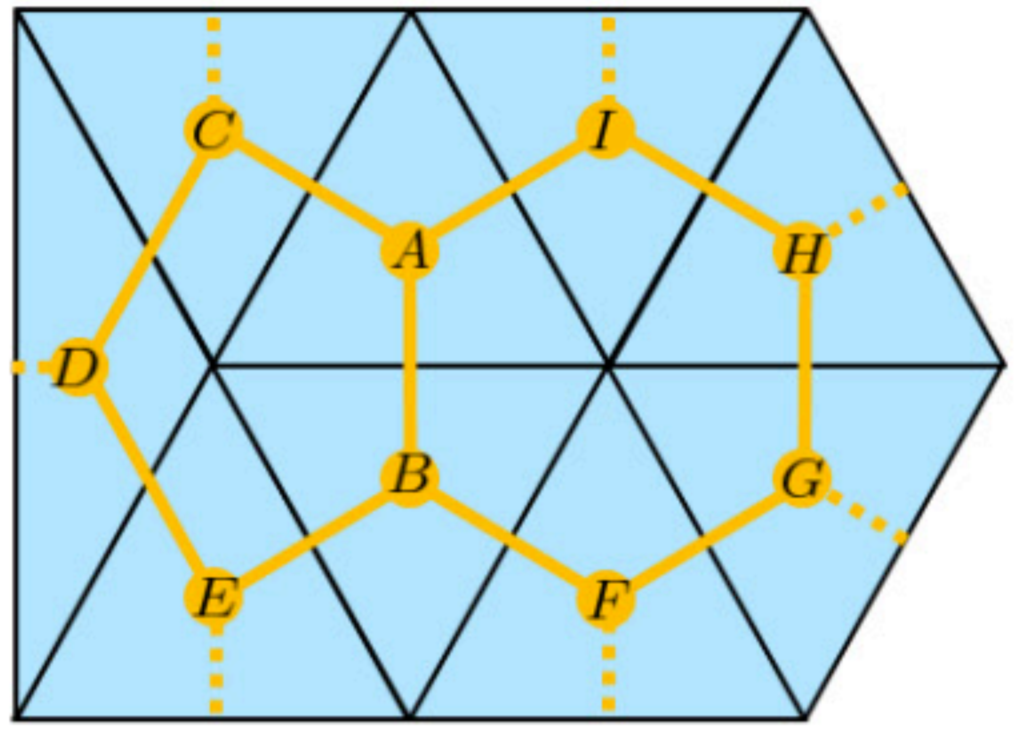


Deep Geometric Texture Synthesis [SIGGRAPH 2020]

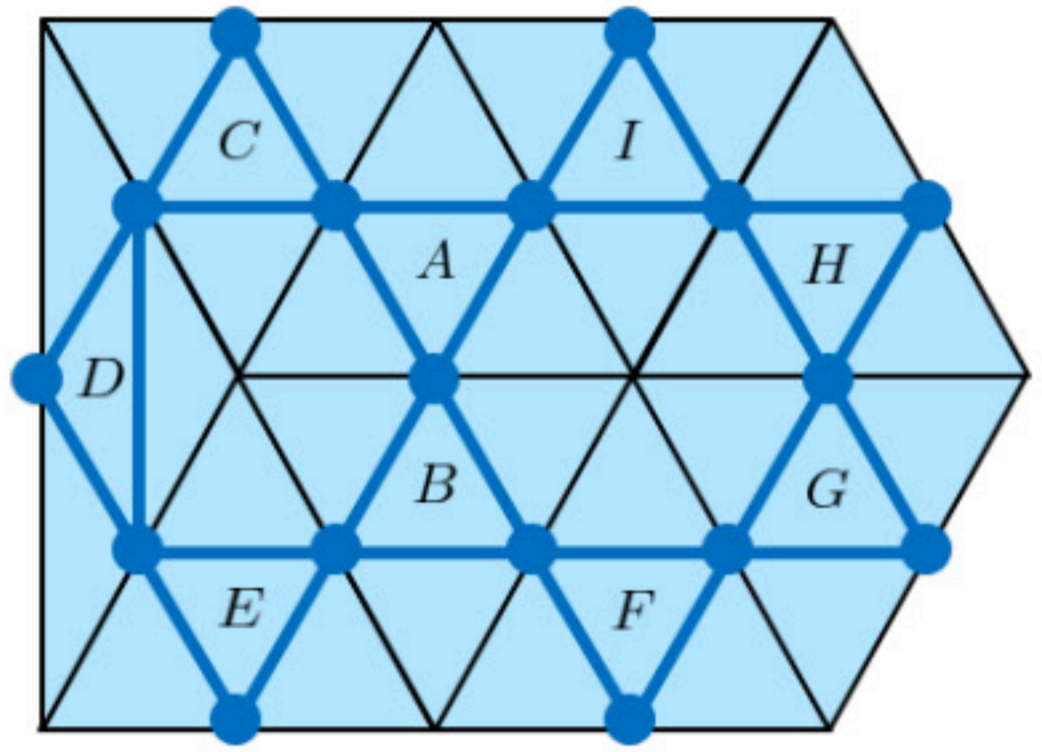
Convolutions on primal/dual mesh graphs



Mesh



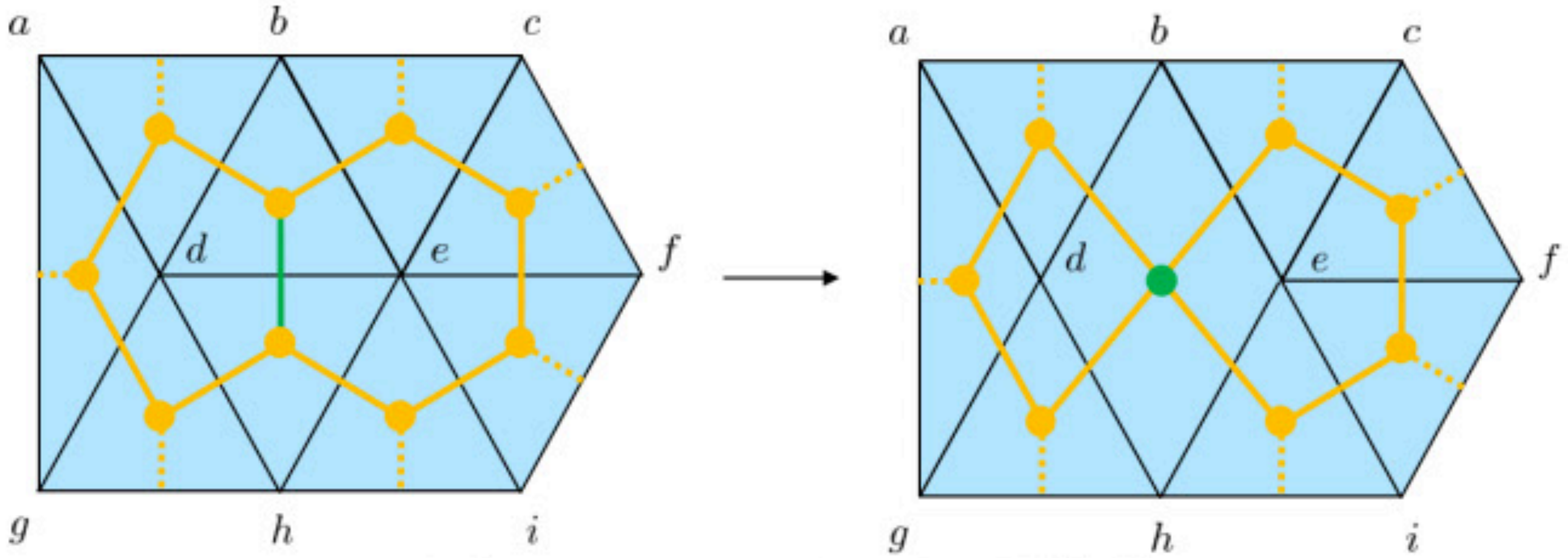
Primal Graph



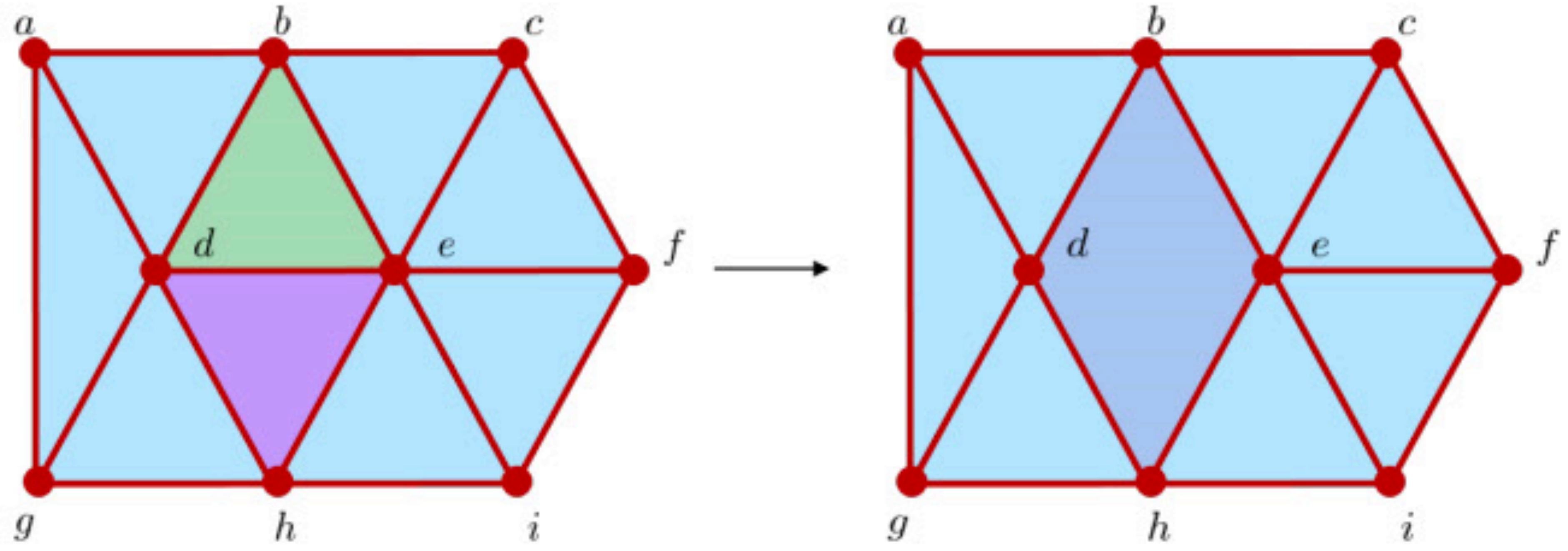
Dual Graph

Primal-Dual Mesh Convolutional Neural Network. Milano et. al [NeurIPS 2020]

Mesh Pooling via Edge Contraction



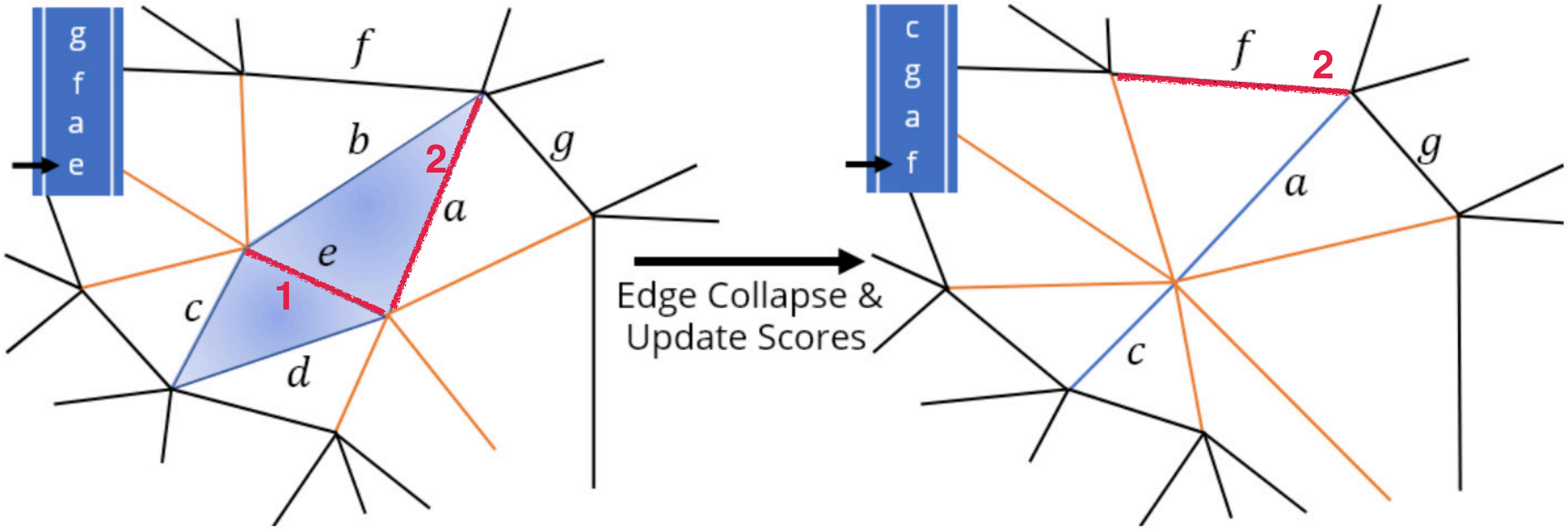
Edge contraction in $\mathcal{P}(\mathcal{M})$



Face merge in $\mathcal{M}$

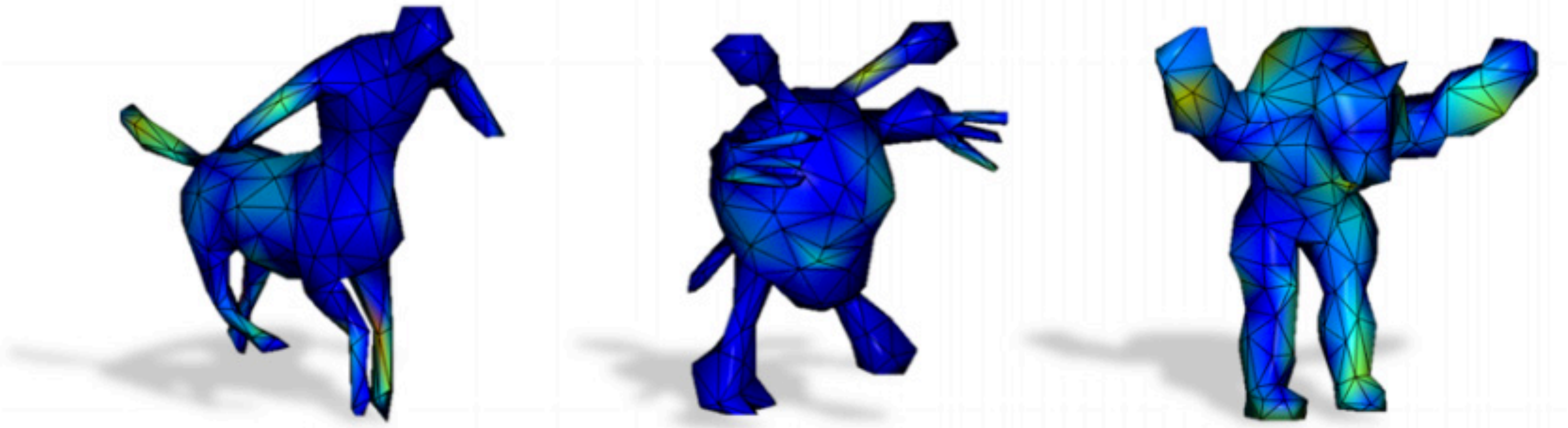
Primal-Dual Mesh Convolutional Neural Network. Milano et. al [NeurIPS 2020]

Enhanced mesh pooling

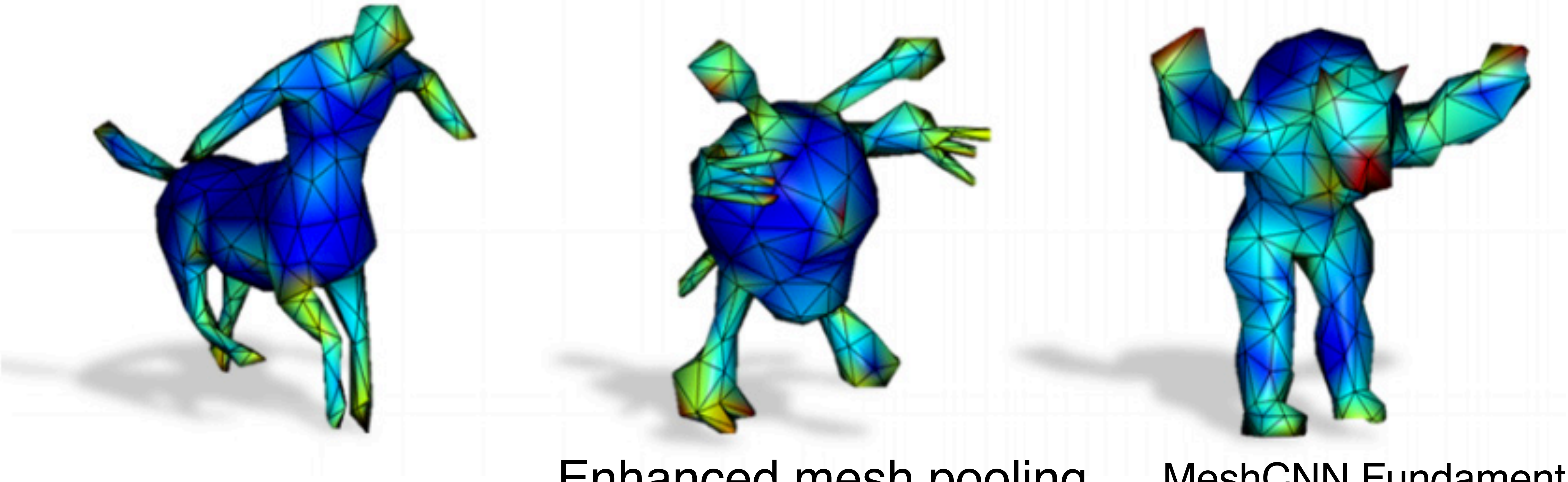


MeshCNN Fundamentals. Barda et. al [ArXiv 2021]

Enhanced mesh pooling



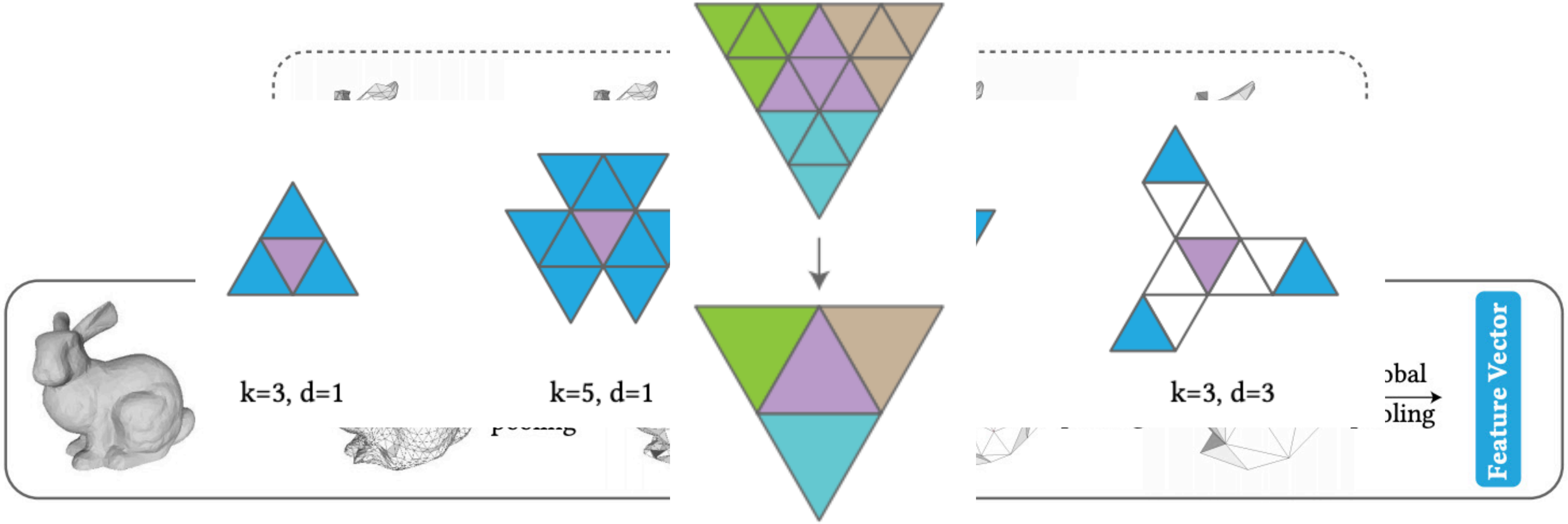
Original mesh pooling



Enhanced mesh pooling

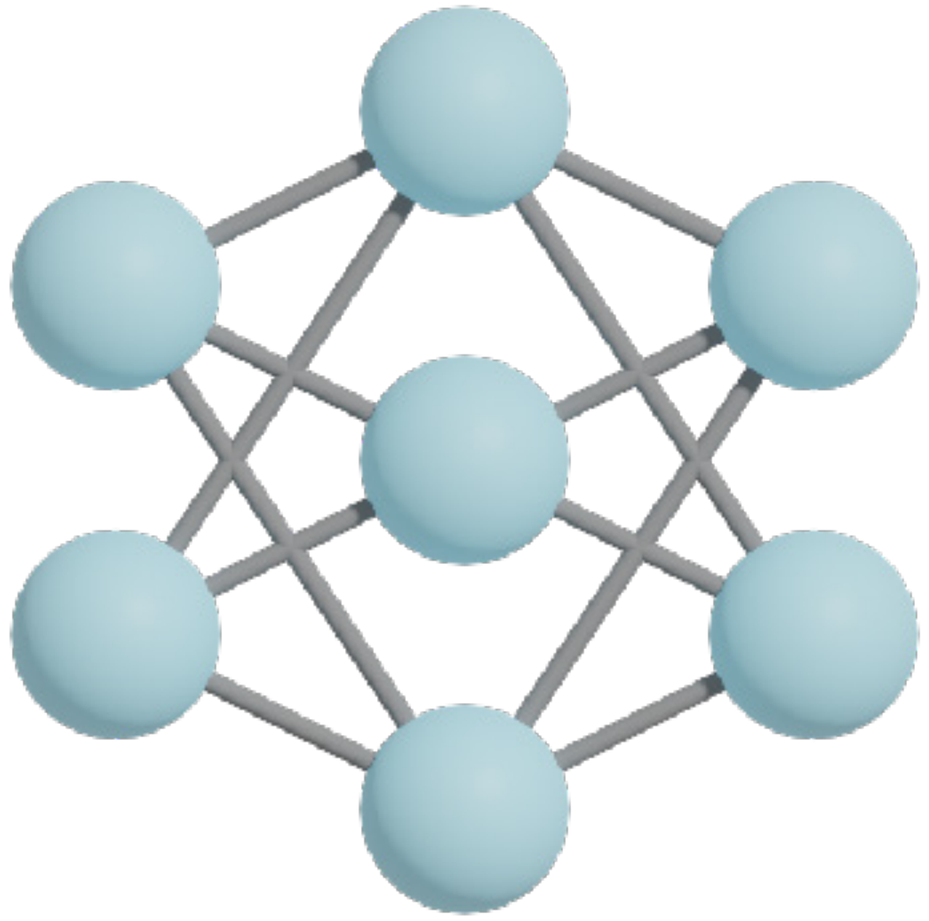
MeshCNN Fundamentals. Barda et. al [ArXiv 2021]

Convolution and pooling from subdivision



Subdivision-based Mesh Convolutional Neural Networks. Hu et. al [ArXiv 2021]

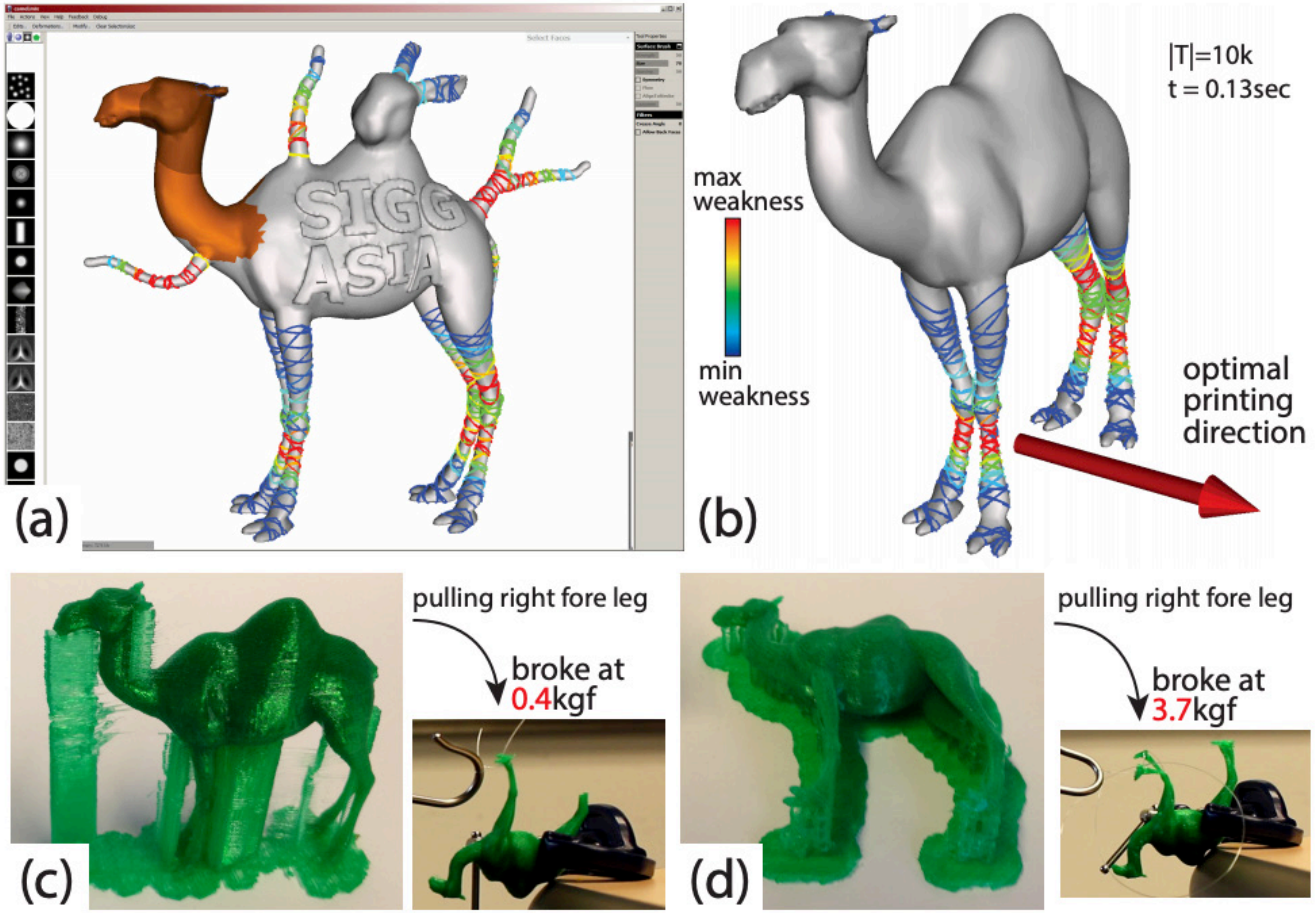
Invariance to rigid transformations



still
It’s a bunny

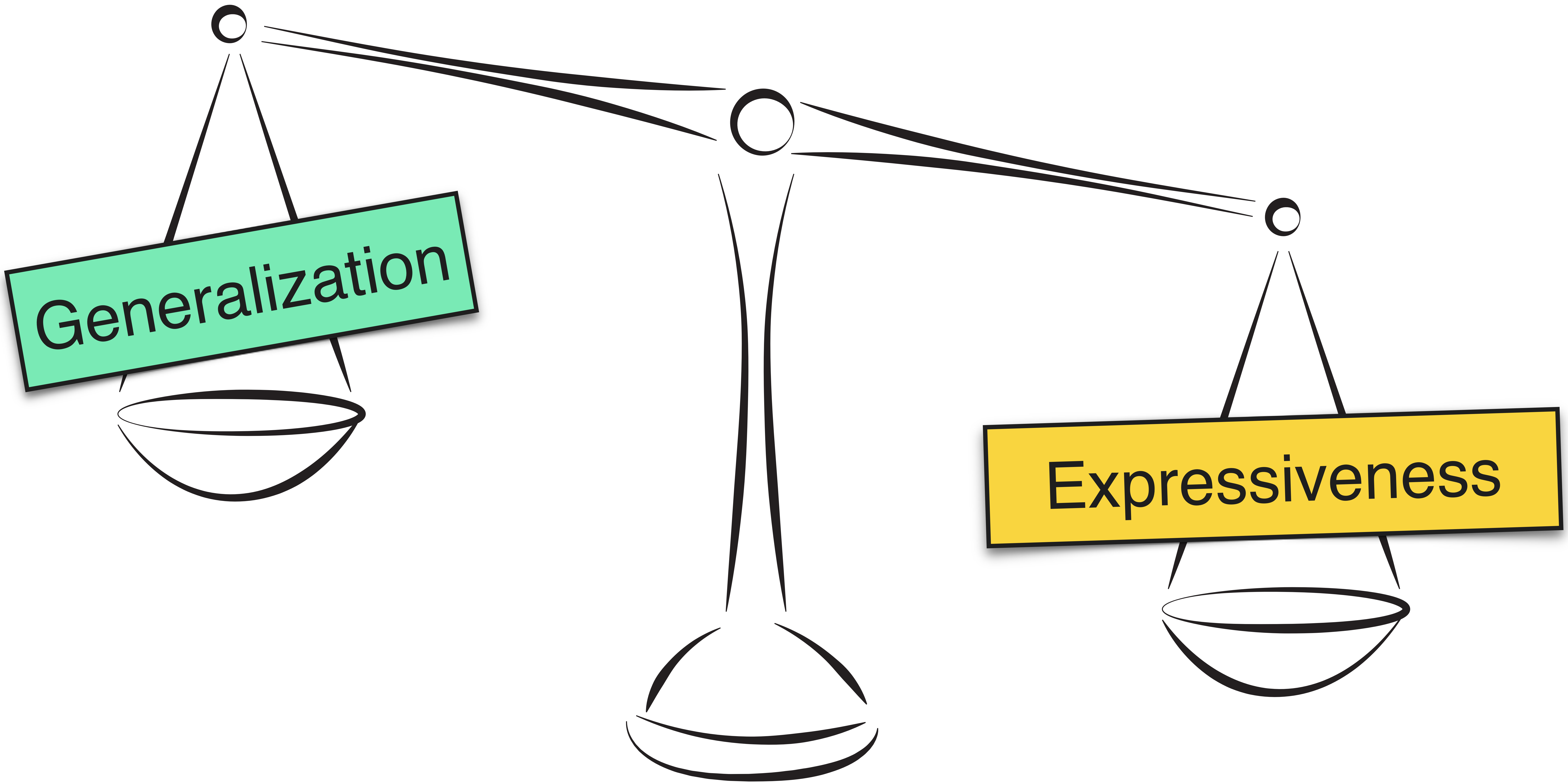
Invariance to rigid transformations

… sometimes this can be too restrictive

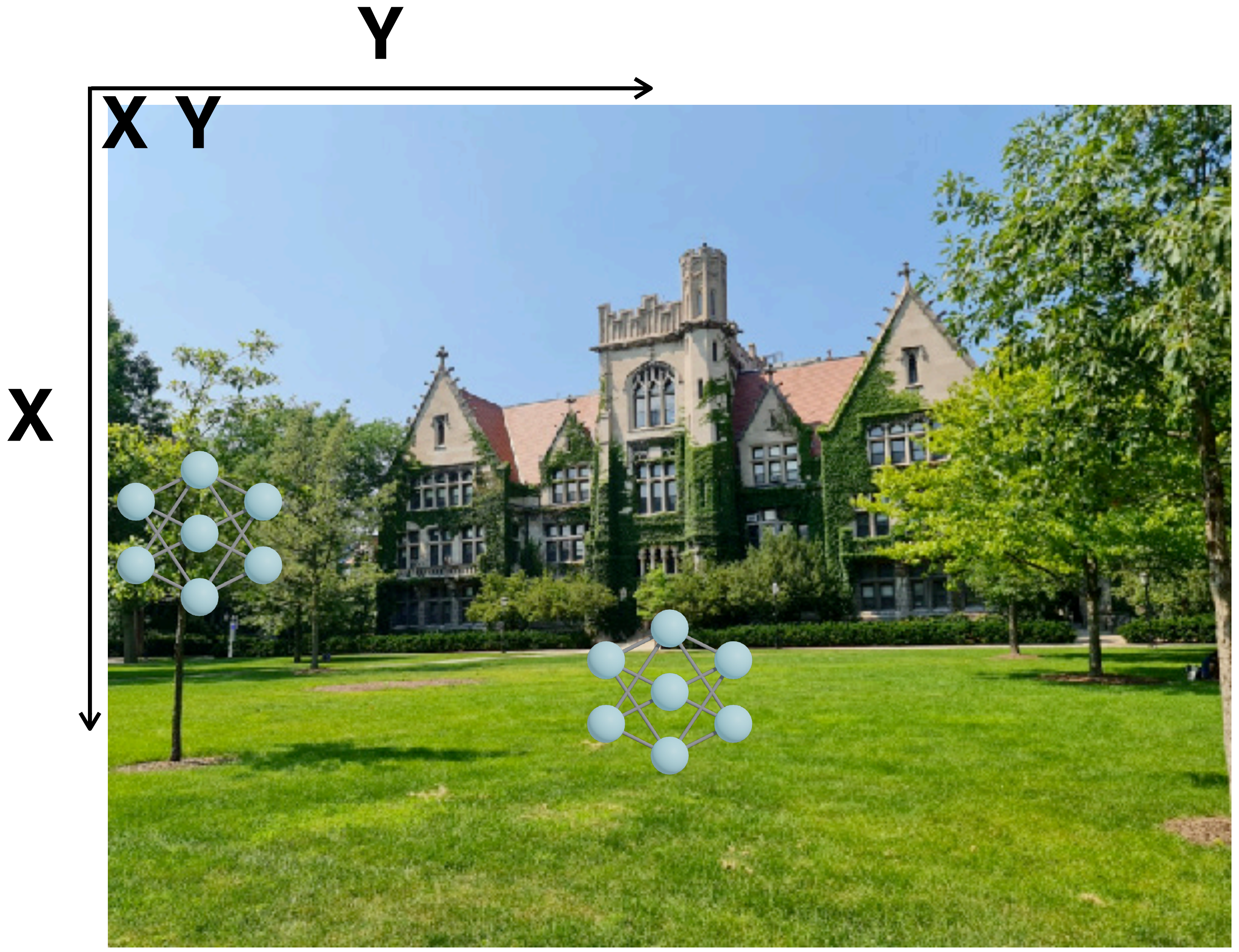


Umetani & Schmidt [SIGGRAPH Asia 2013]

Expressiveness vs. Generalization

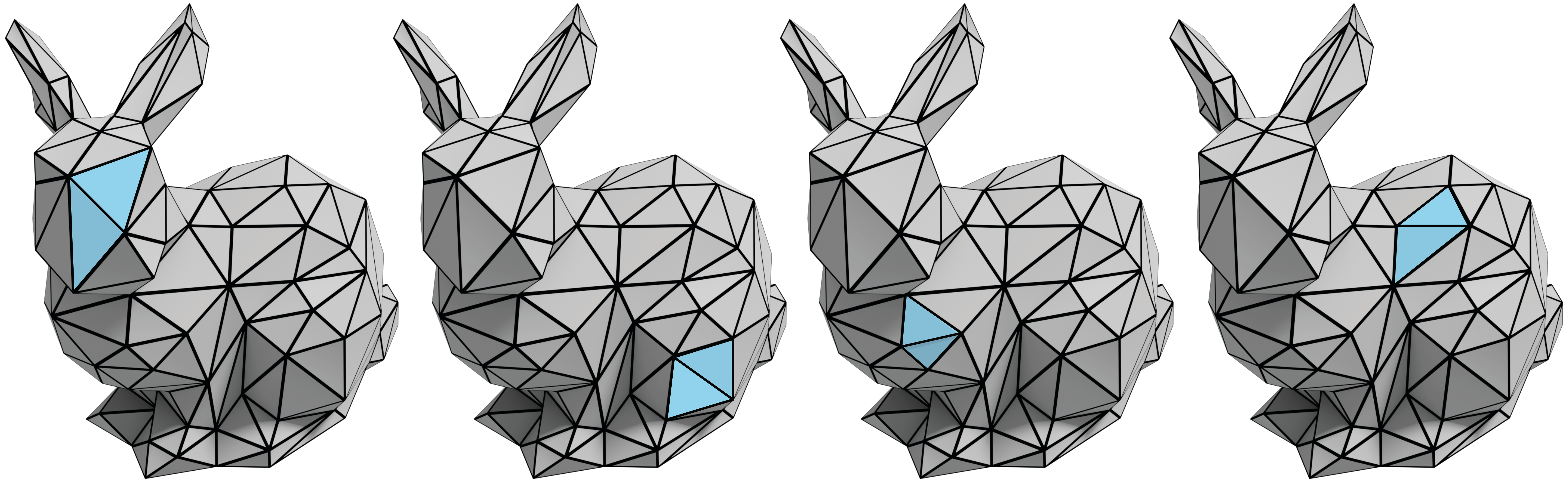


Break shift-invariance



Gain expressive power

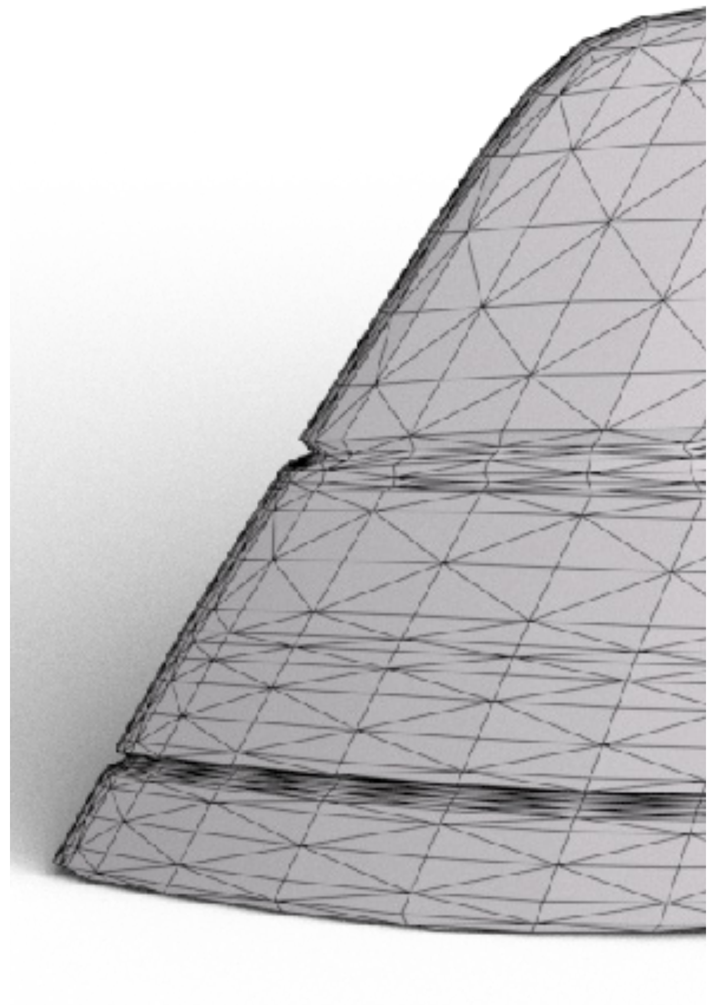
Helps better fit the data

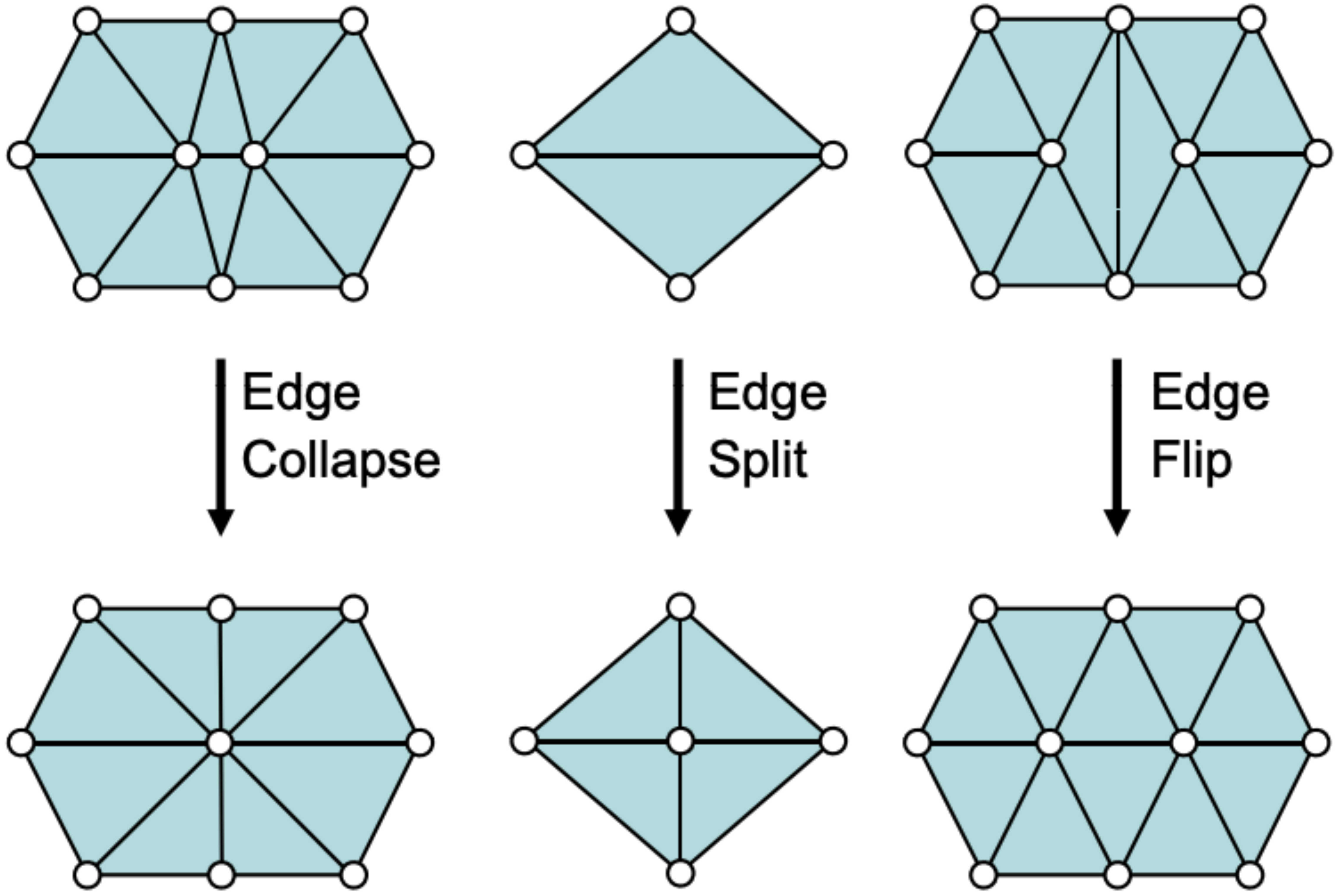


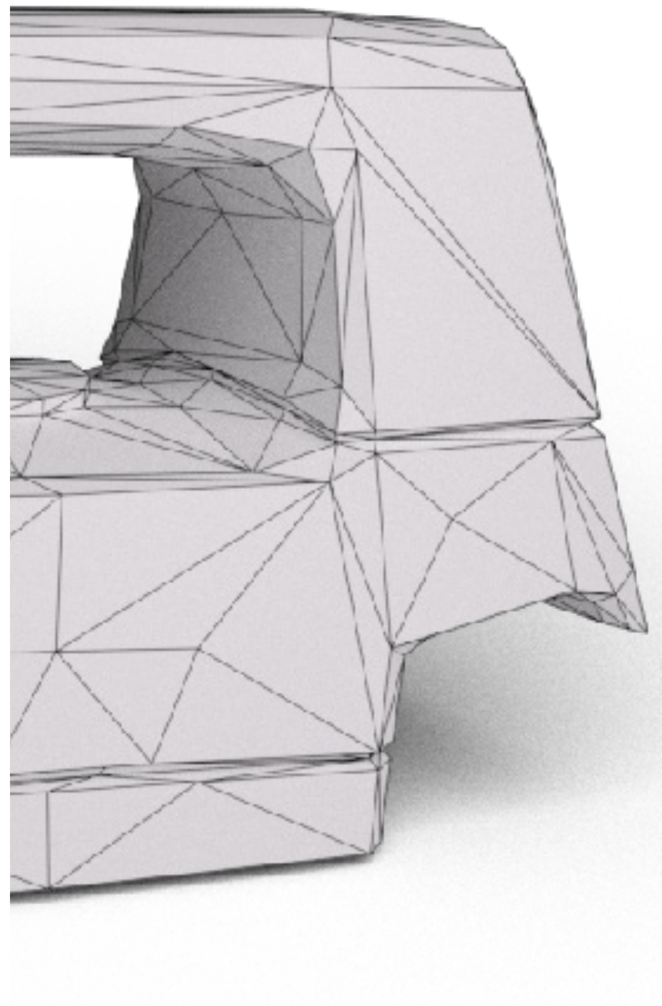
Fast Fourier Features Tancik et. al [2020]

Triangulation robustness

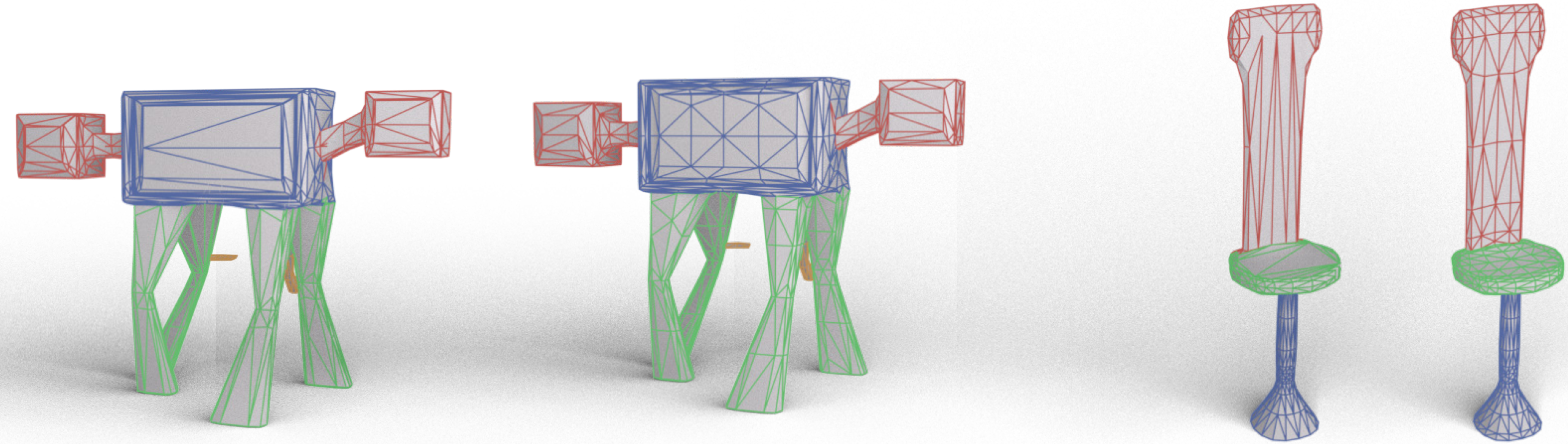
Perform simple augmentations





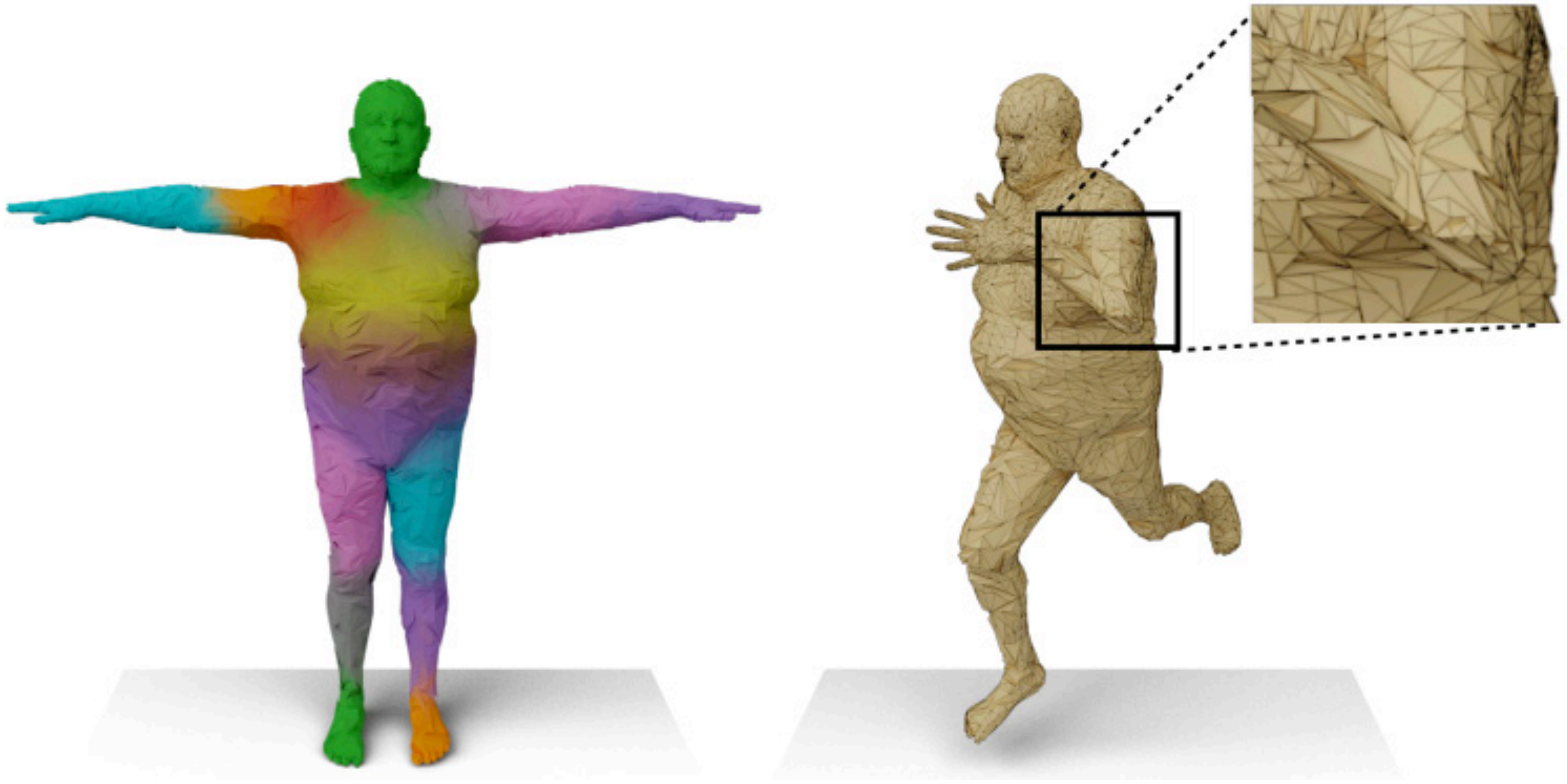


Triangulation robustness in segmentation

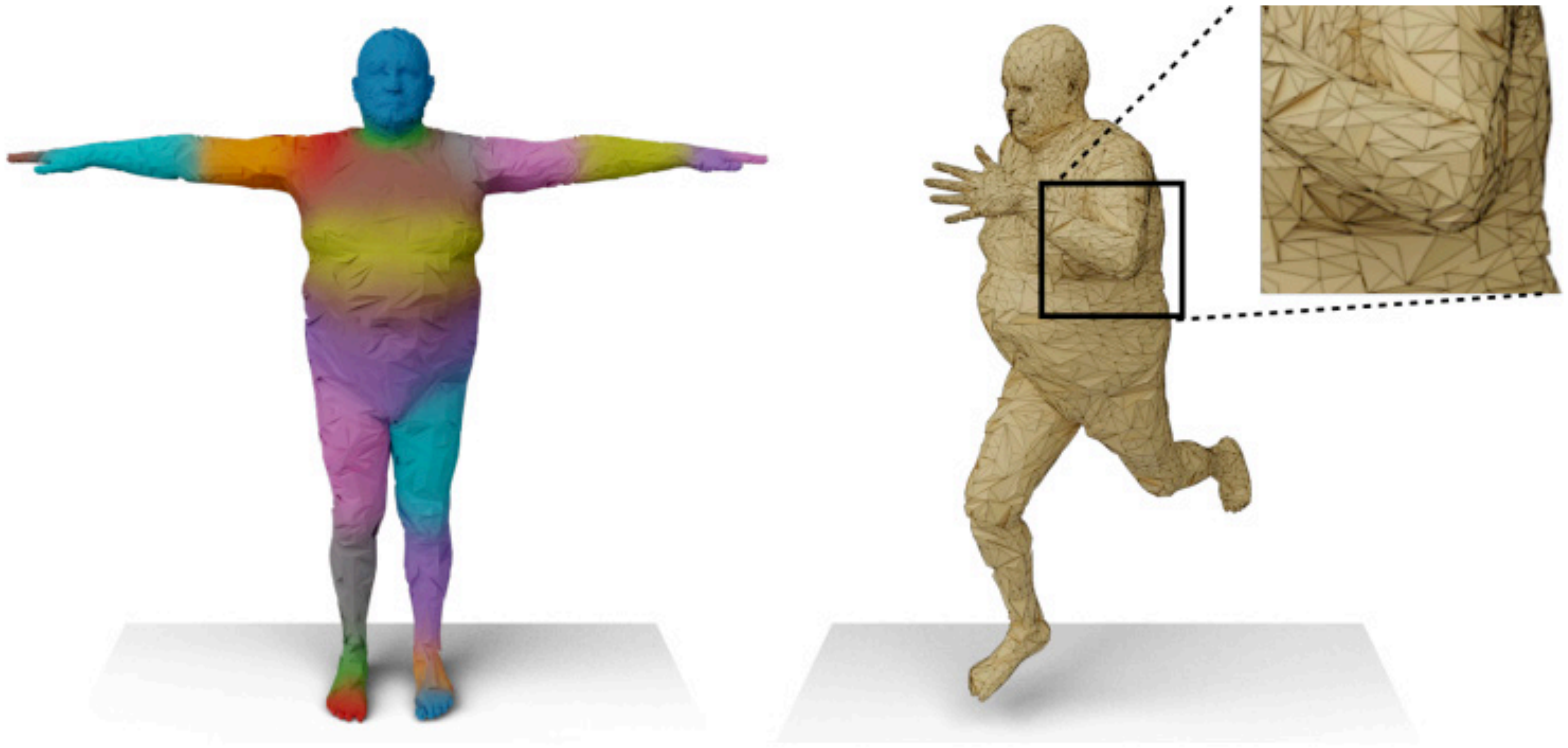


MeshCNN: a Network with an Edge [SIGGRAPH 2019]

Triangulation robustness in deformation



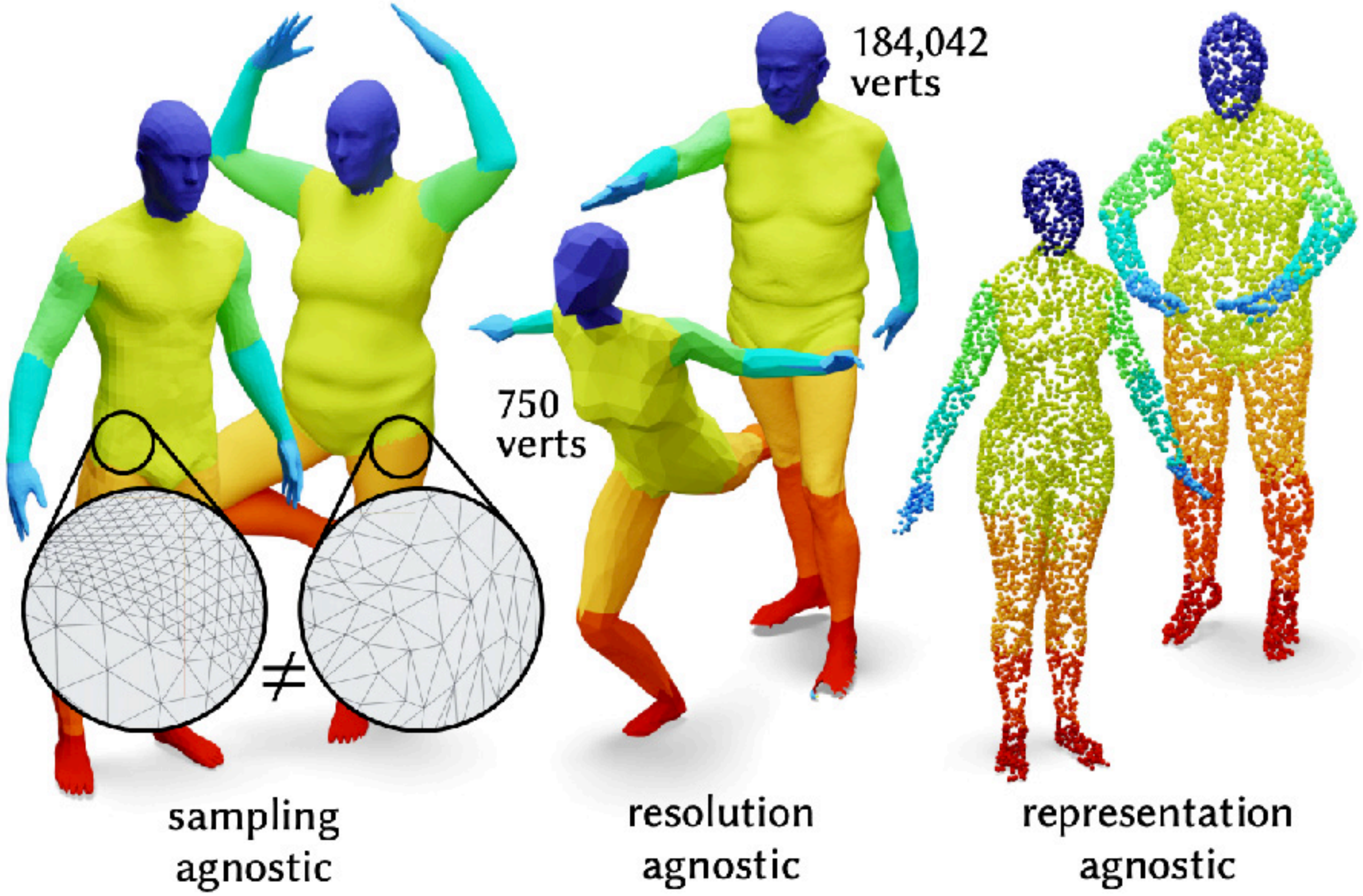
Without augmentation



With augmentation

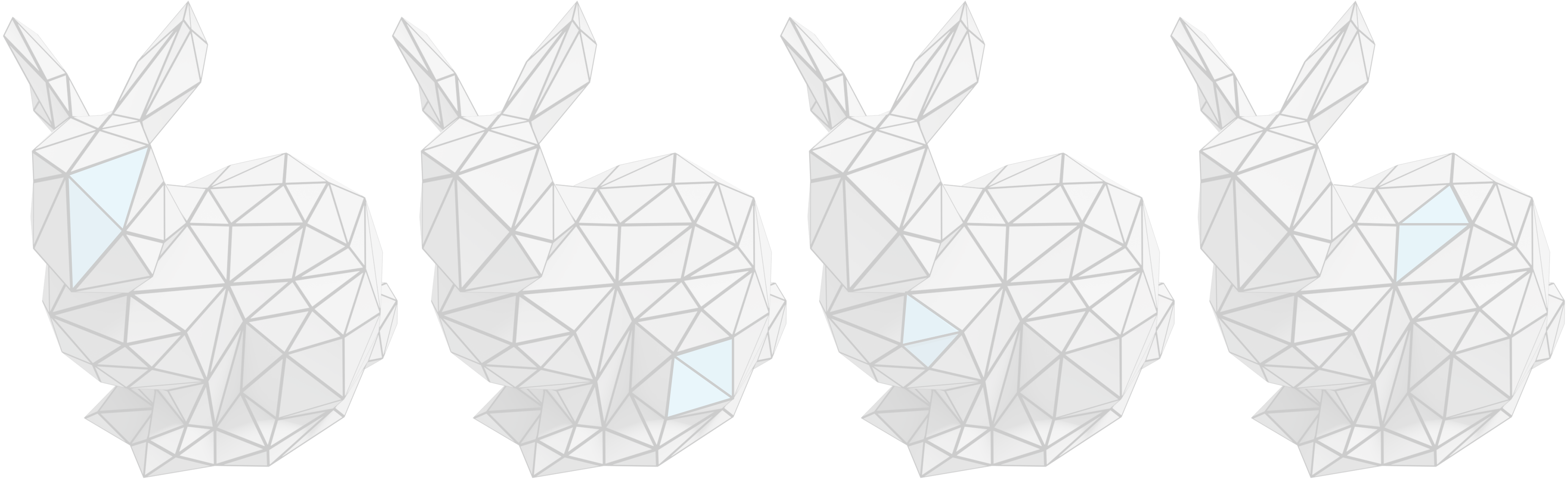
Neural Blend Shapes [SIGGRAPH 2021]

Triangulation robustness via diffusion



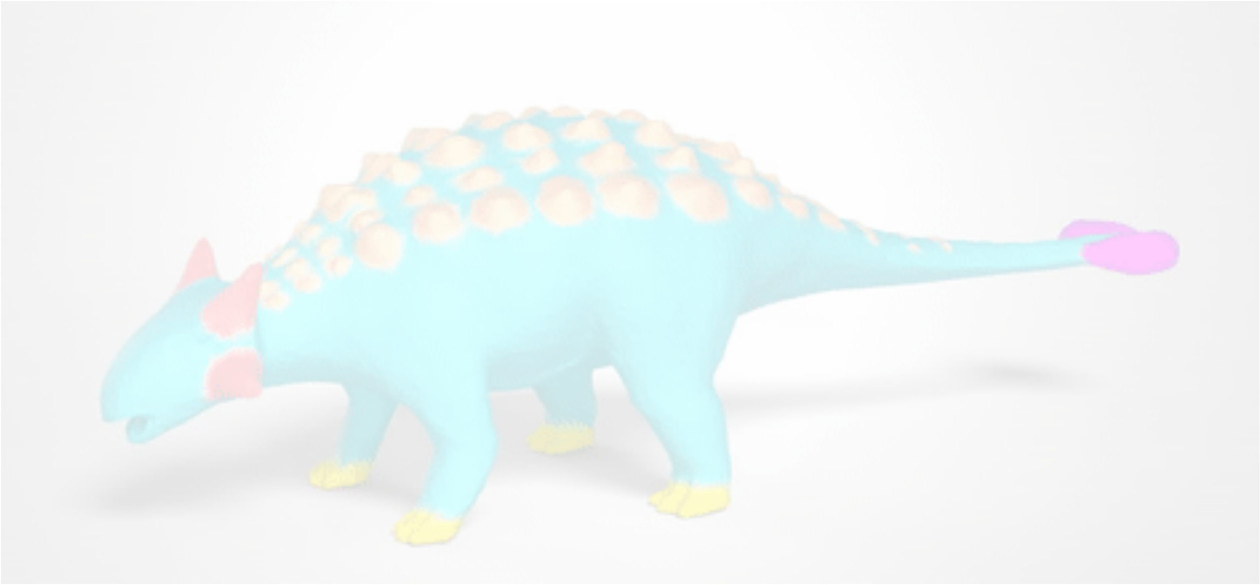
DiffusionNet. Sharp et. al [ArXiv 2021]

Mesh Convolutional Neural Networks

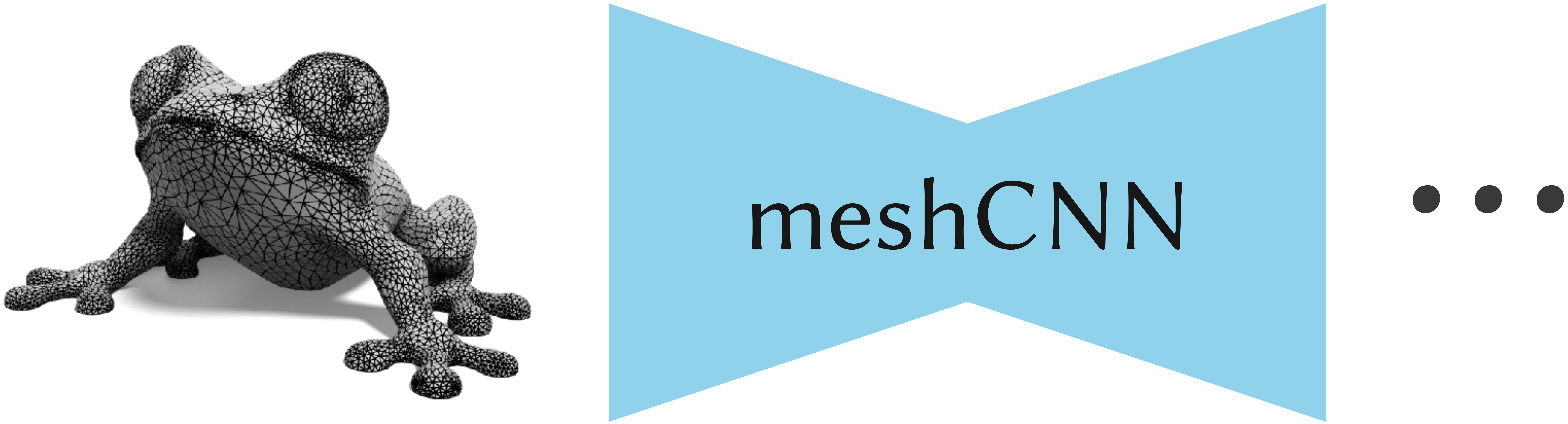


Machine Learning & Geometry Processing

Learning from a Single Mesh



A Fundamental Tool: MeshCNN



meshCNN

• • •

A Small Component

loss
meshCNN
build dV
+
sample
dist
C
X

A Small Component

loss
meshCNN
build dV
+
sample
dist
C
X

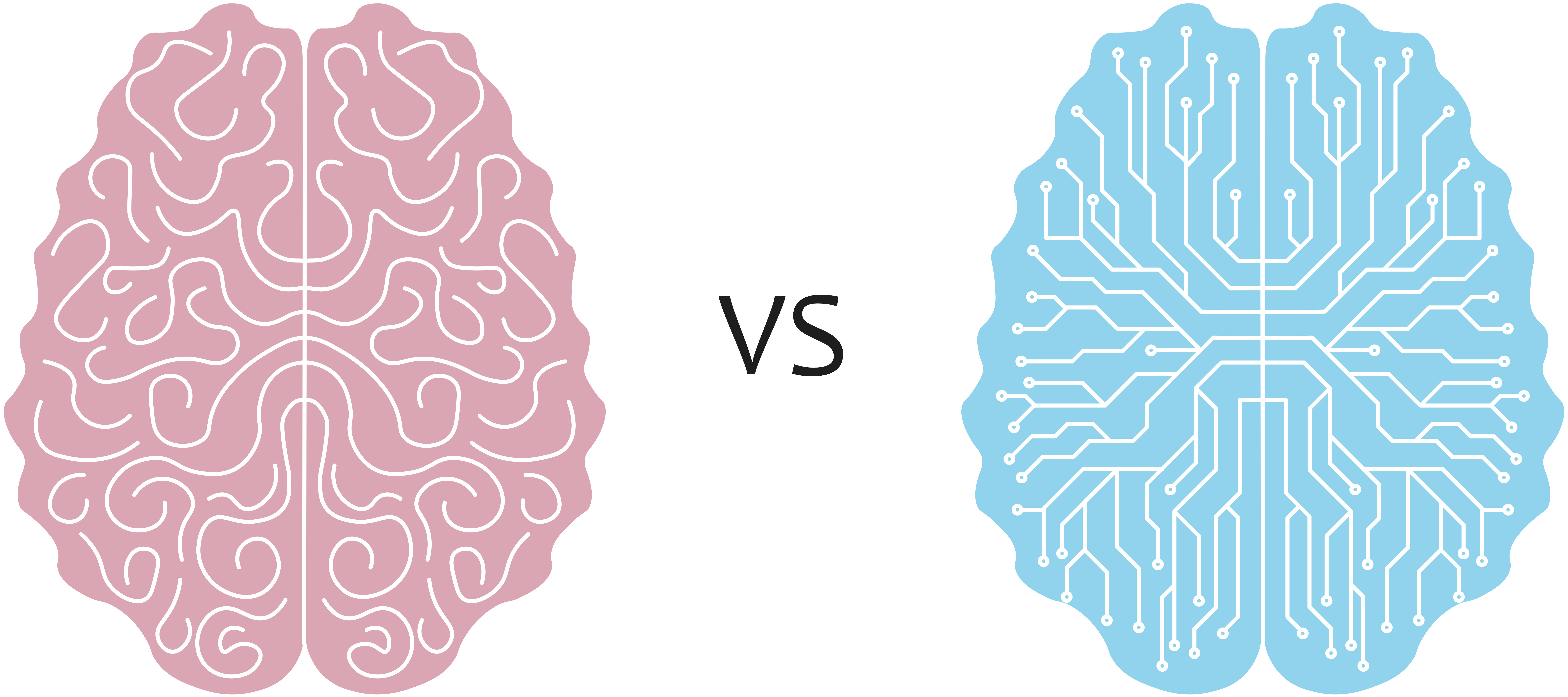

VS

Image Processing

SIFT

[Lowe 1999]

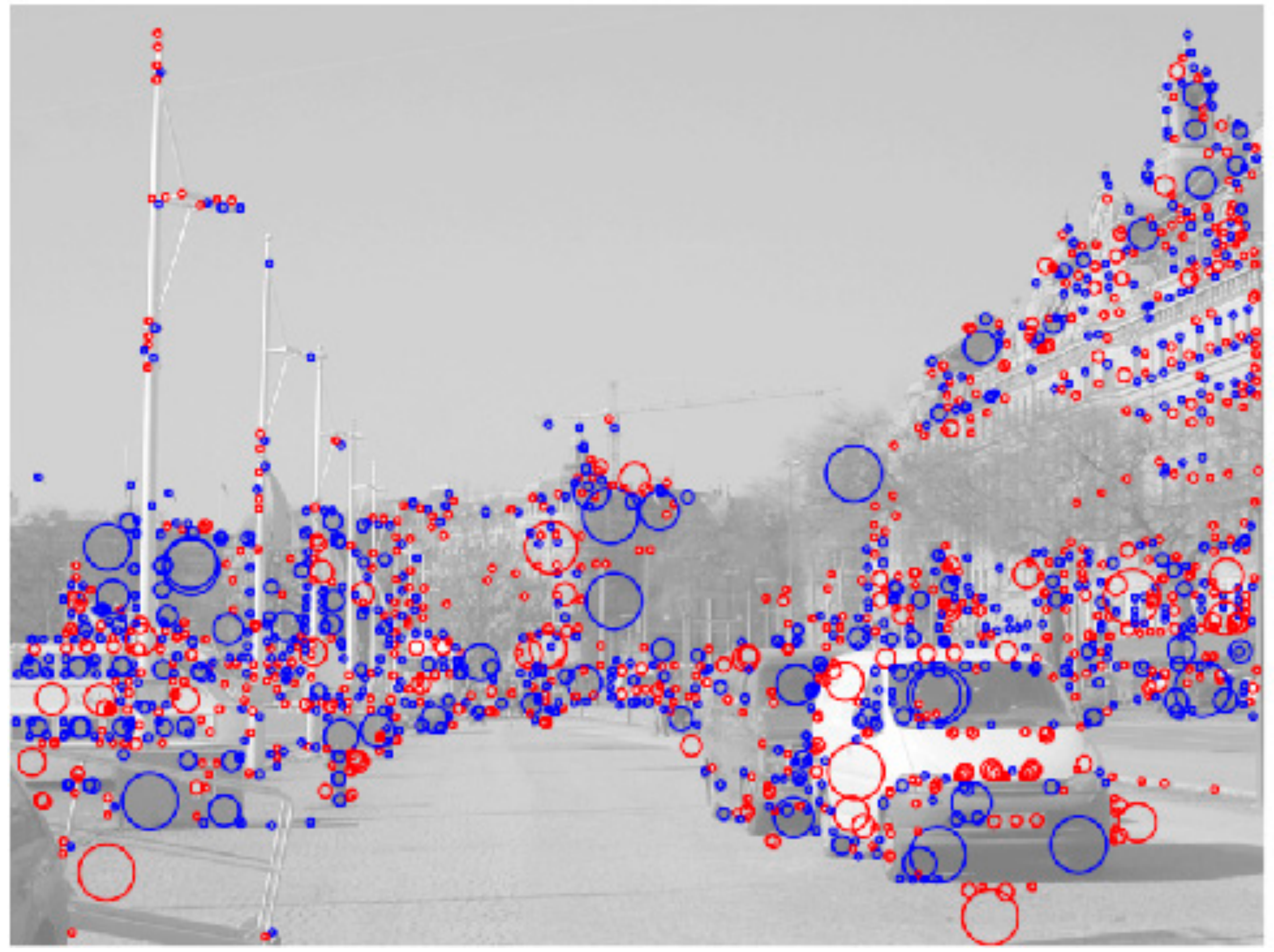
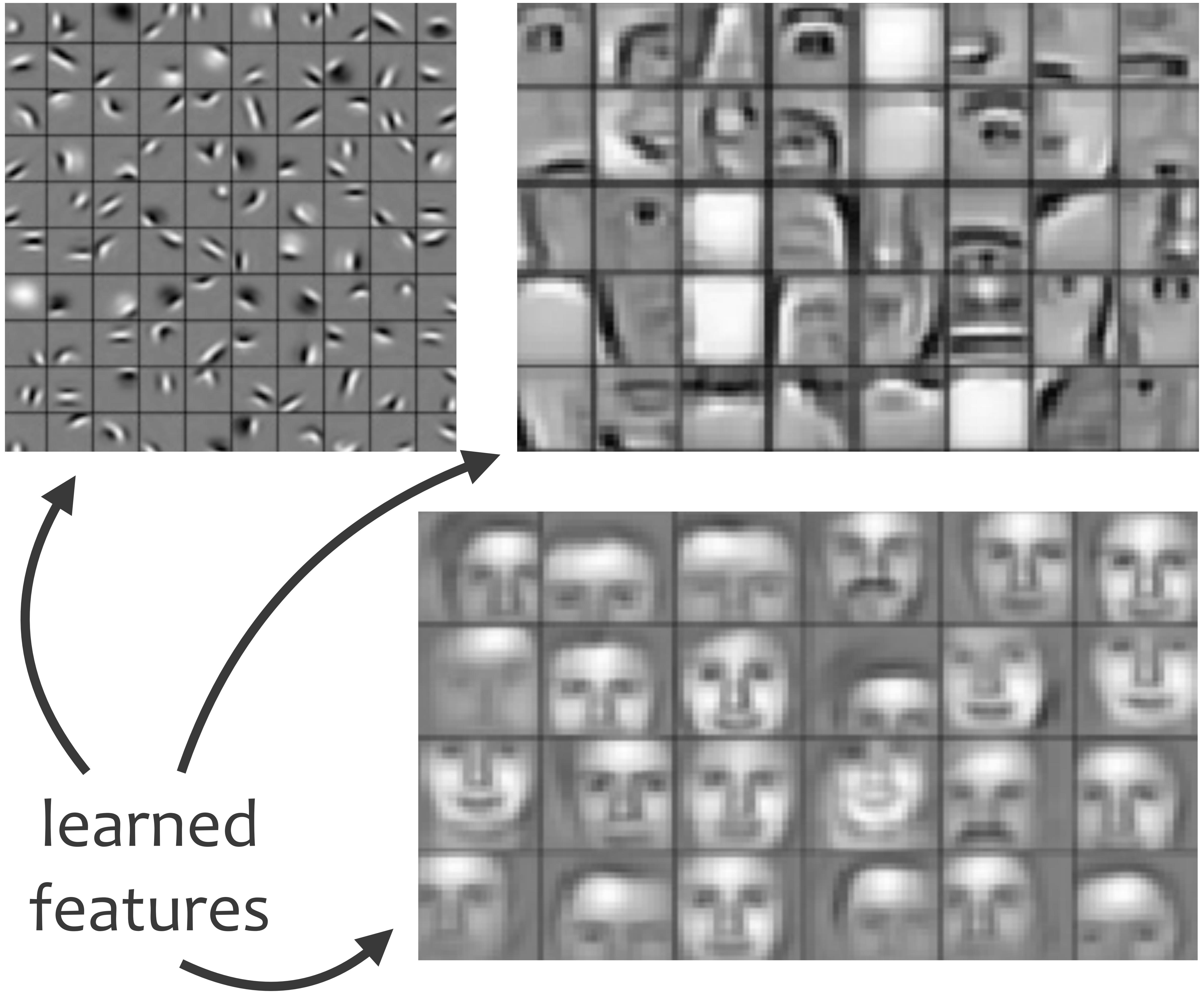


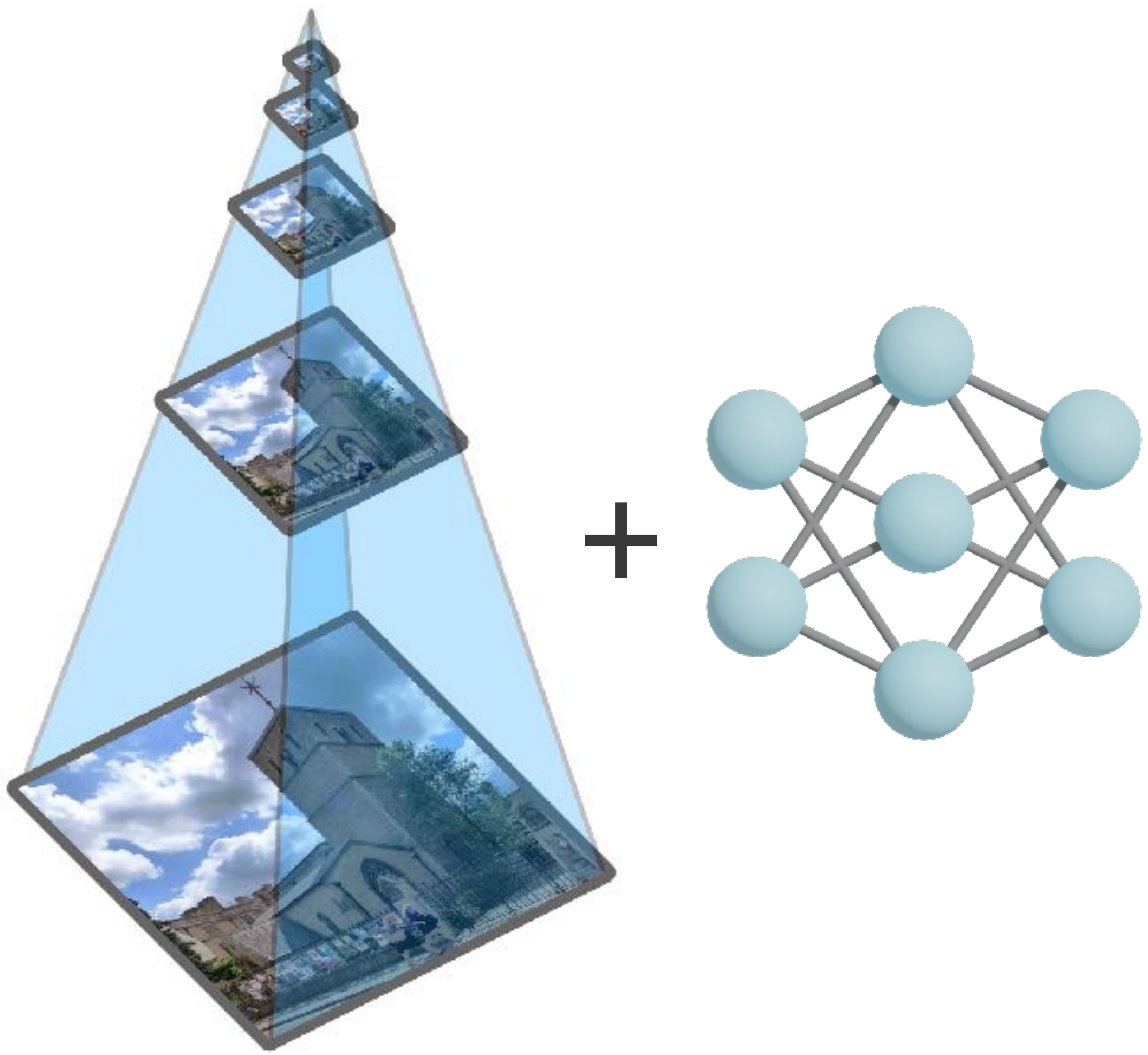
Image Processing

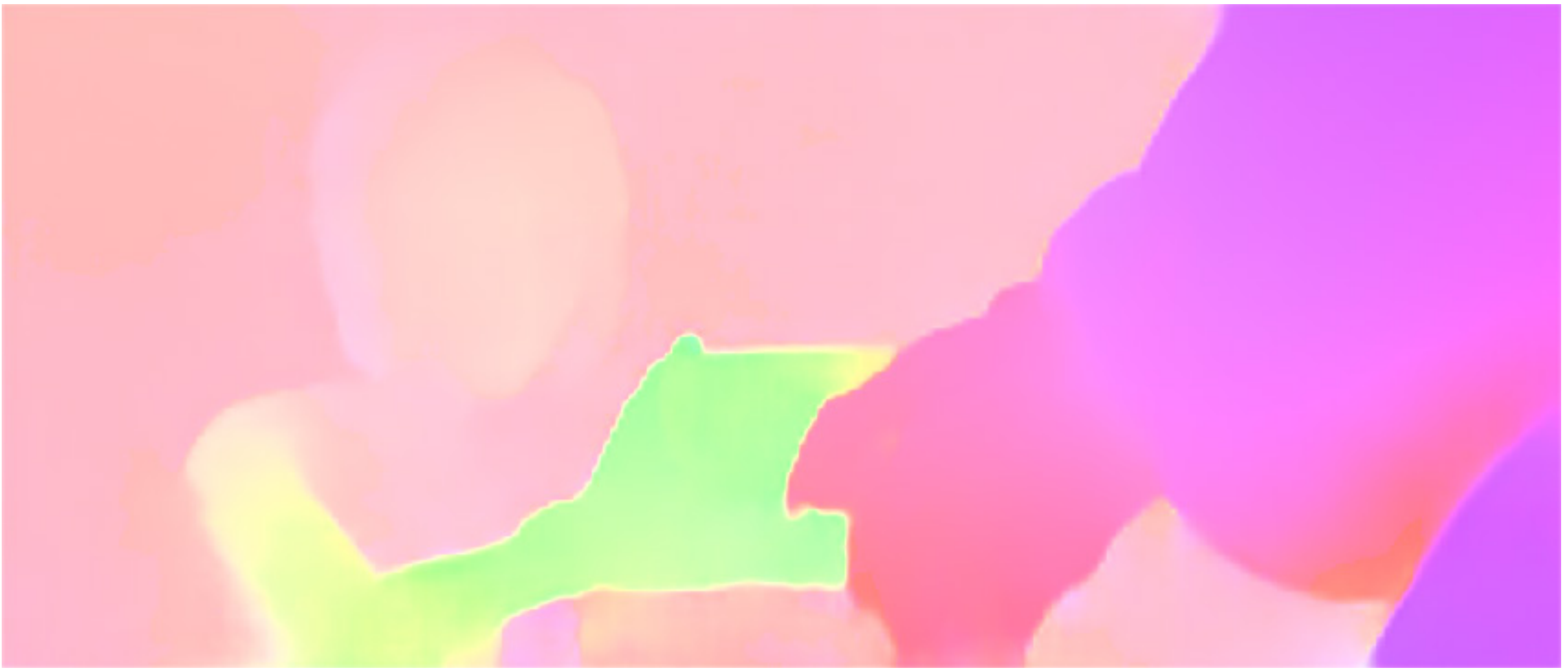
SIFT
[Lowe 1999]



learned features

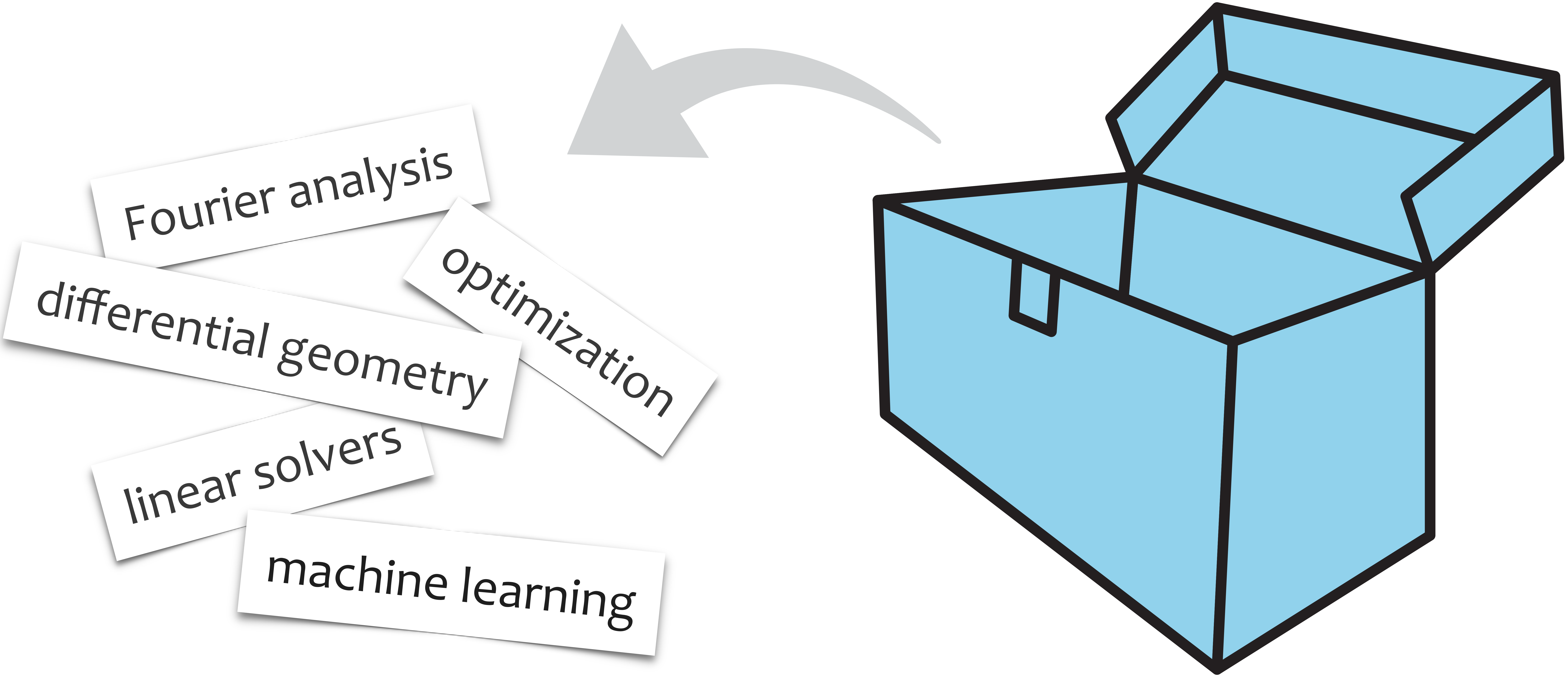
E.g., Optimal Flow via Pyramid





Sun et al. 2018

A tool in our toolbox



Fourier analysis

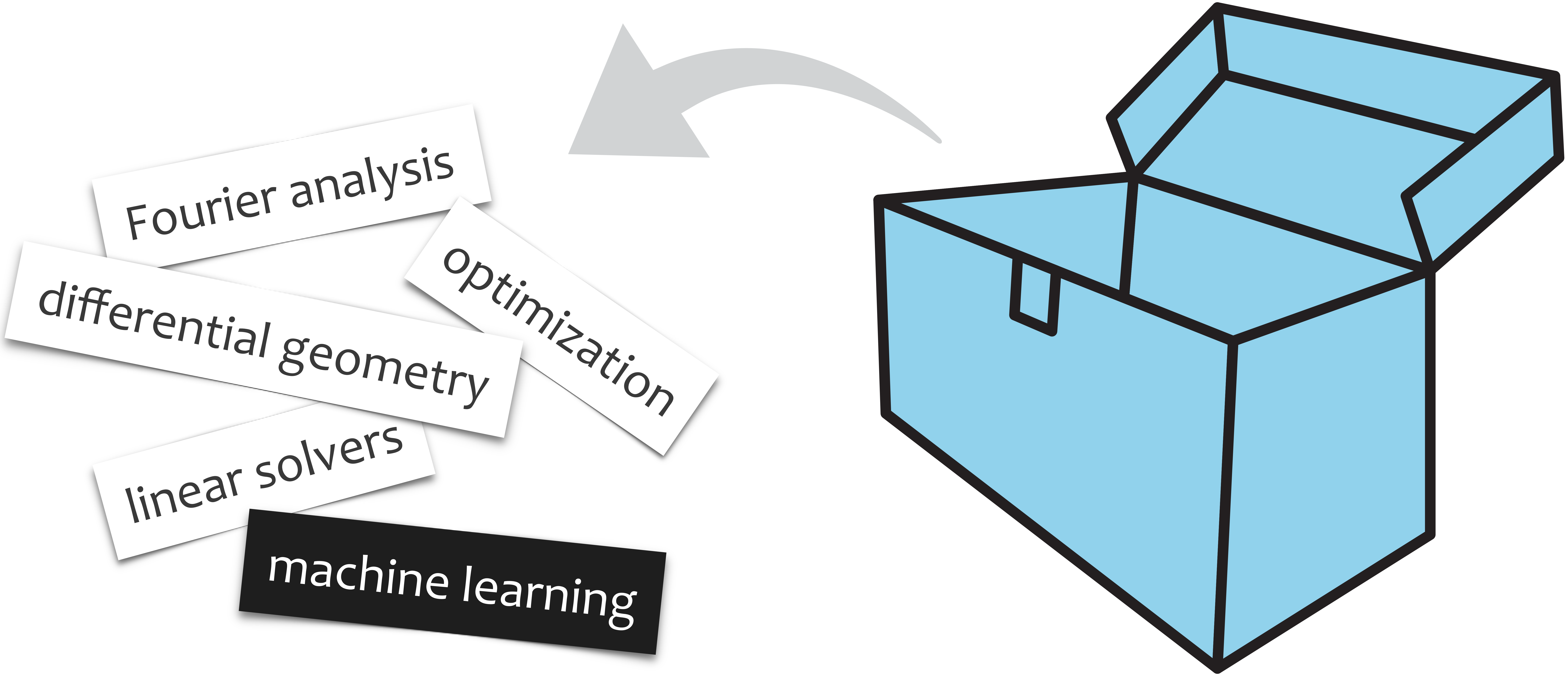
optimization

differential geometry

linear solvers

machine learning

A tool in our toolbox



Fourier analysis

optimization

differential geometry

linear solvers

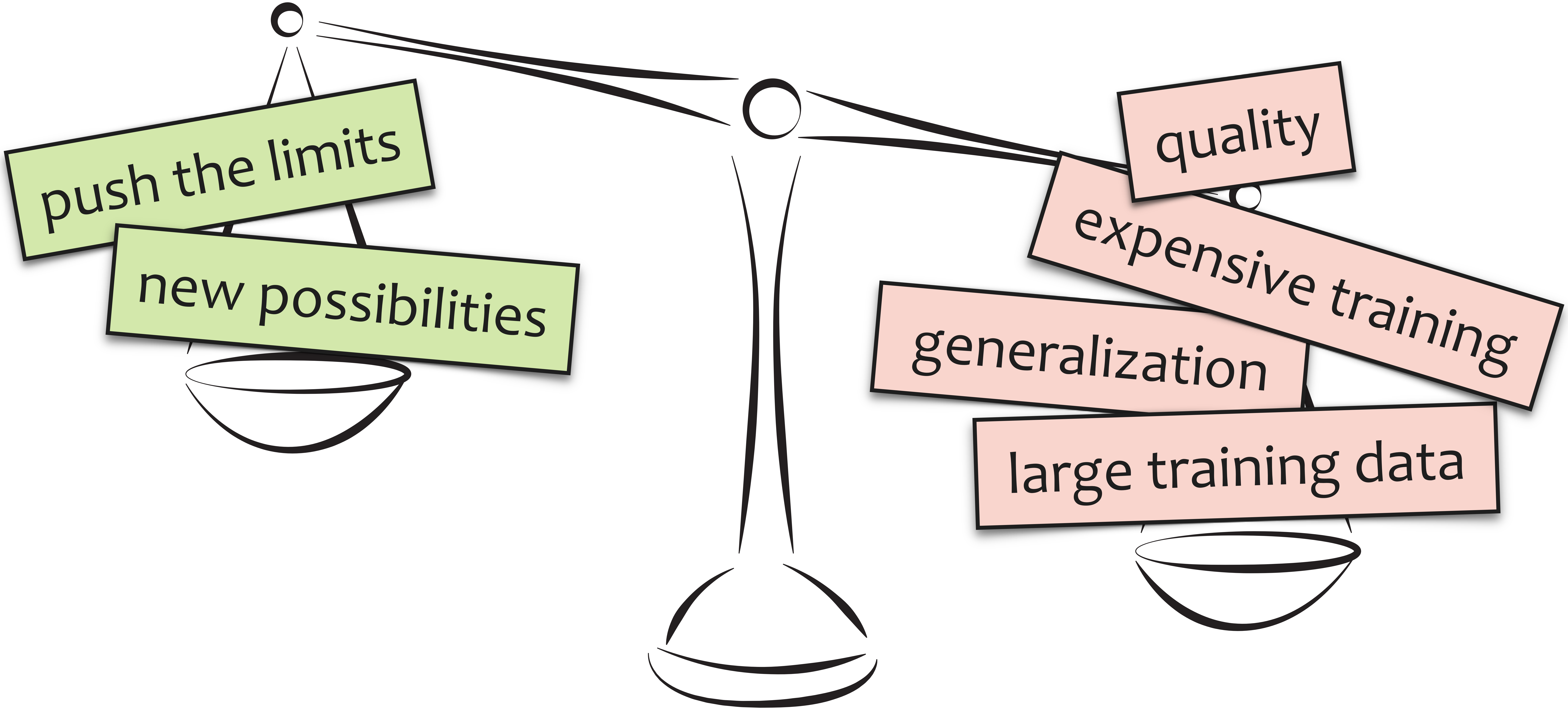
machine learning

No-free-lunch

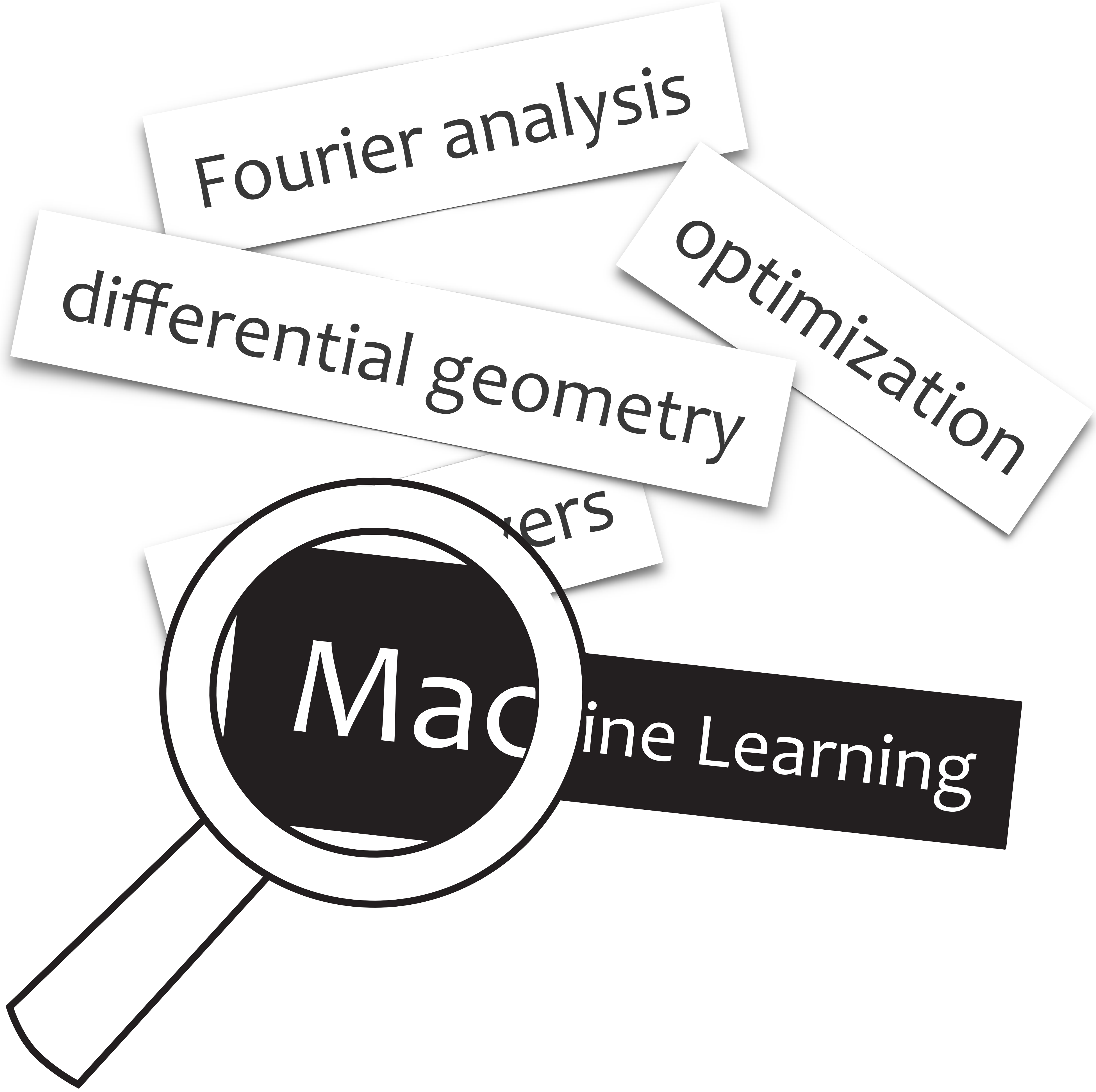
push the limits

new possibilities

No-free-lunch



Overview



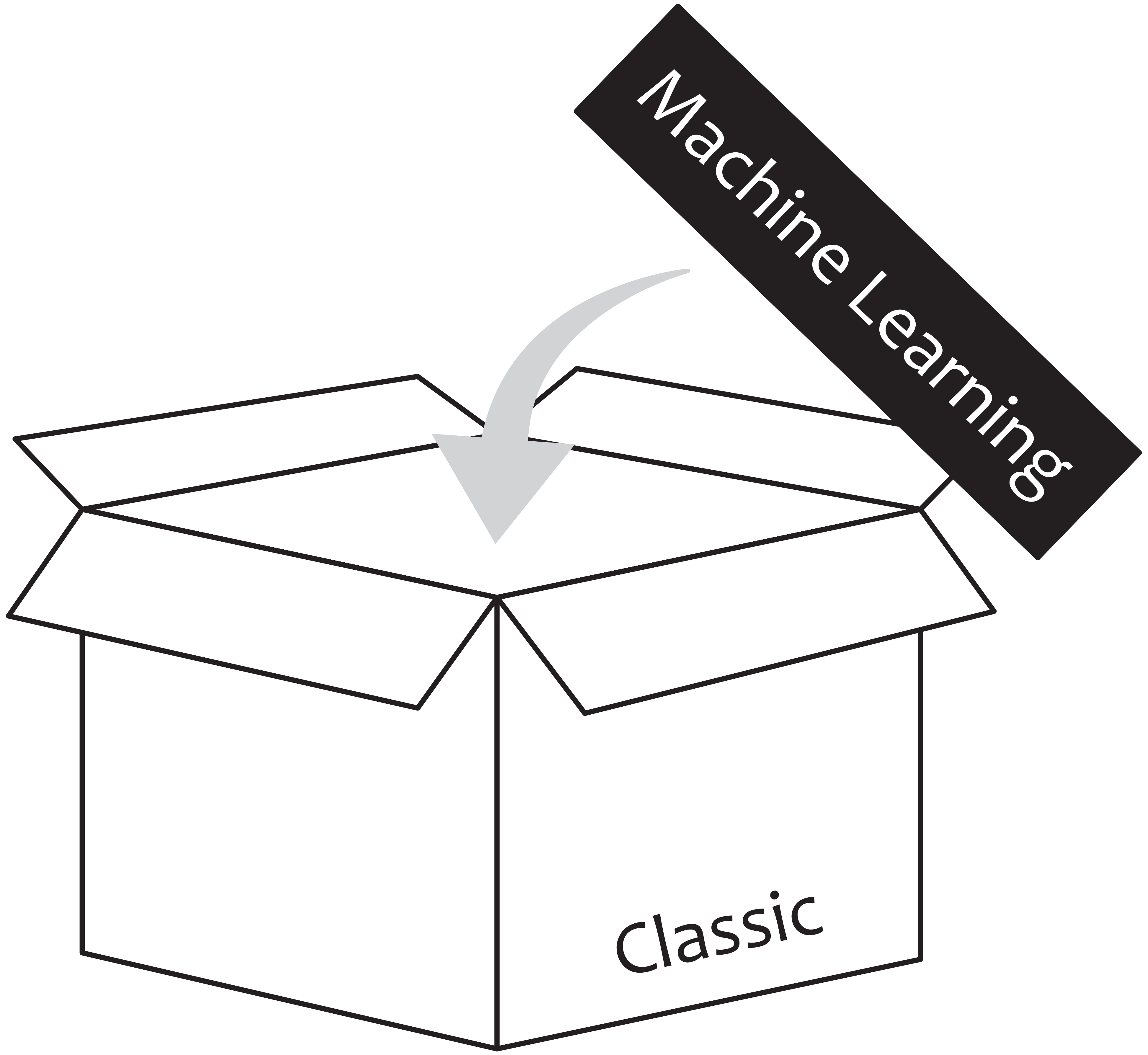
Fourier analysis

differential geometry

optimization

ers

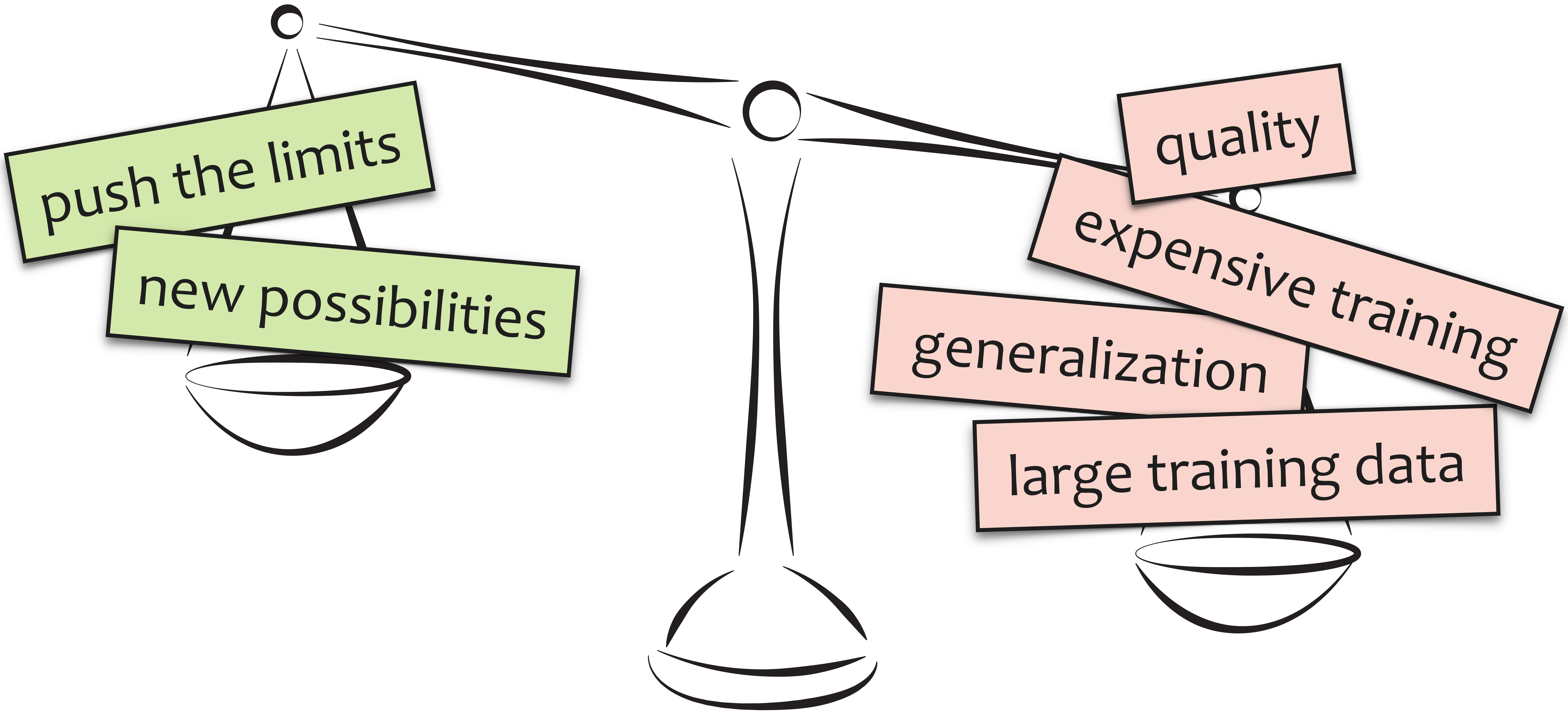
Machine Learning

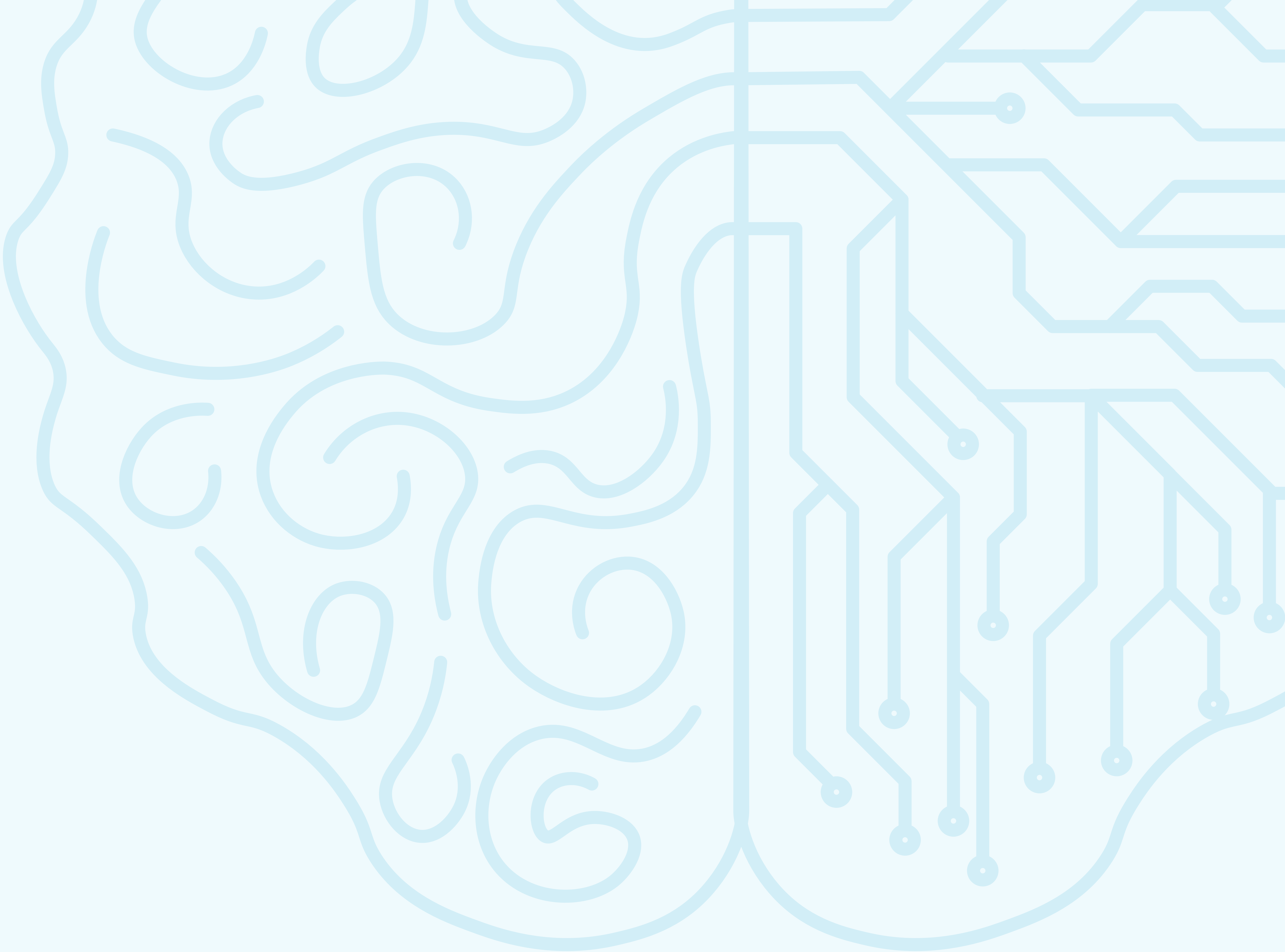


Machine Learning

Classic

Alleviate Limitations



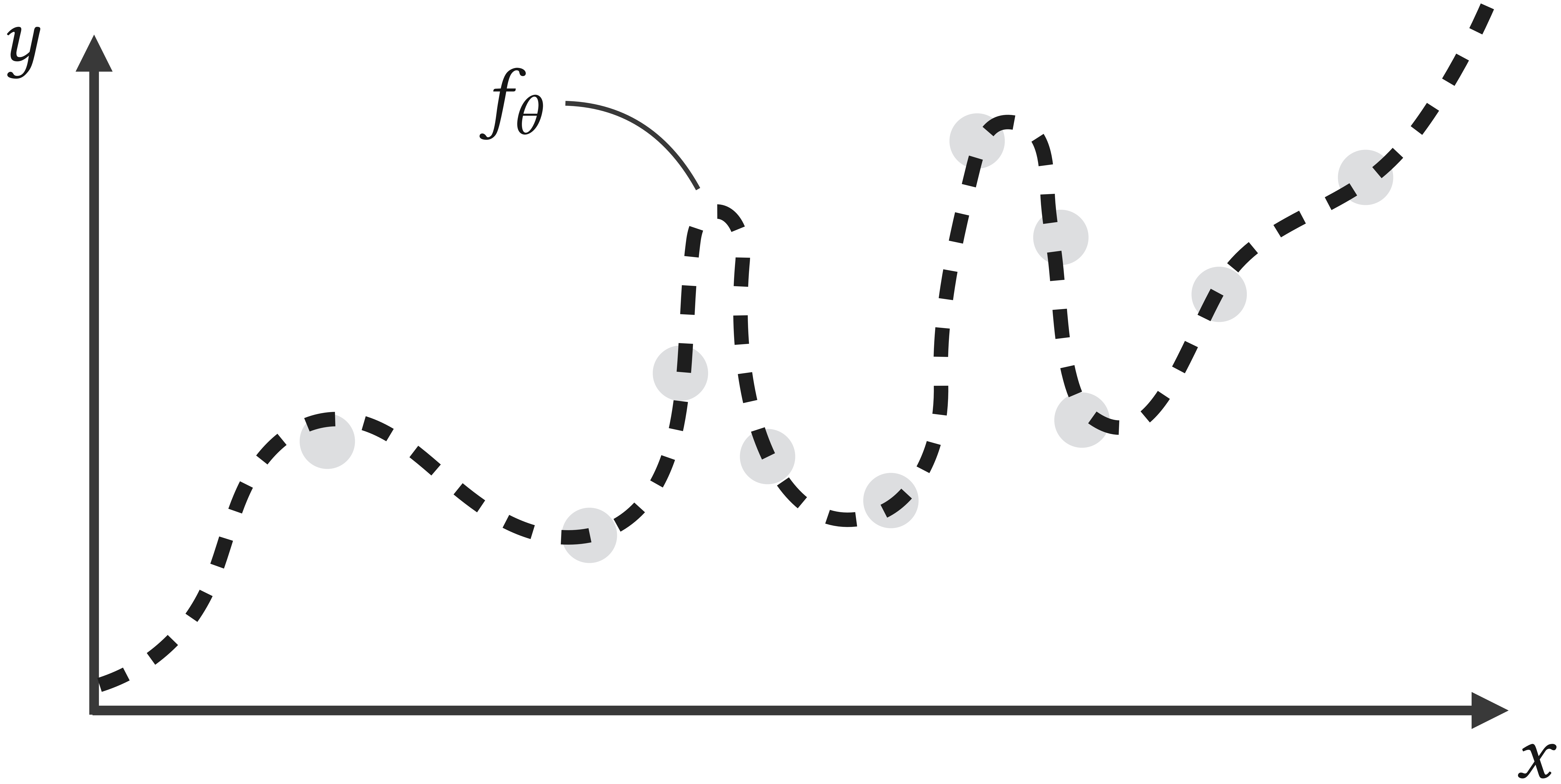


Roles of machine learning

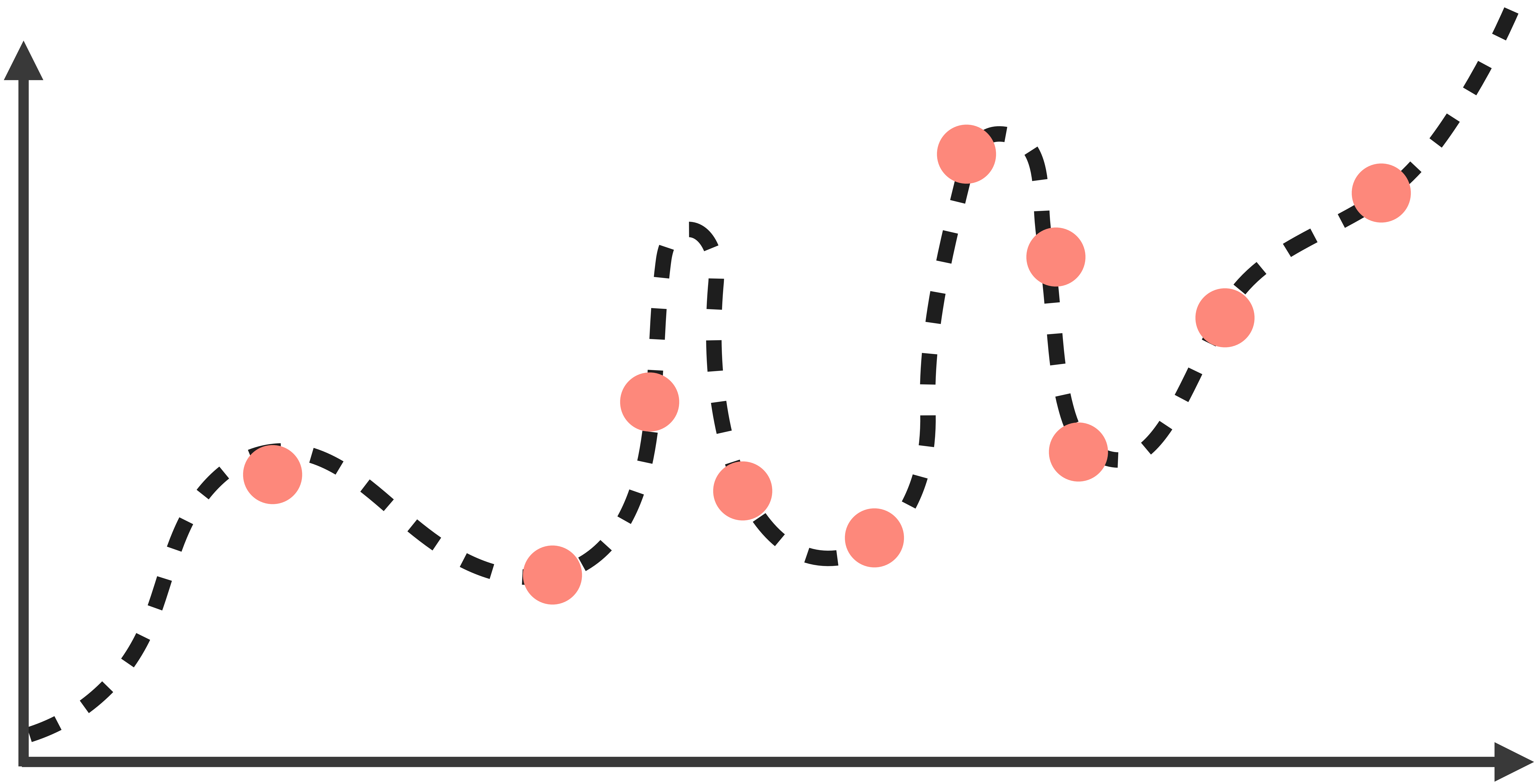
Non-linear Function Approximator

$$\underbrace{y}_{\text{output}} = \underbrace{f}_{\text{ML model}}{}_{\underbrace{\theta}_{\text{network parameters}}}(\underbrace{x}_{\text{input}})$$

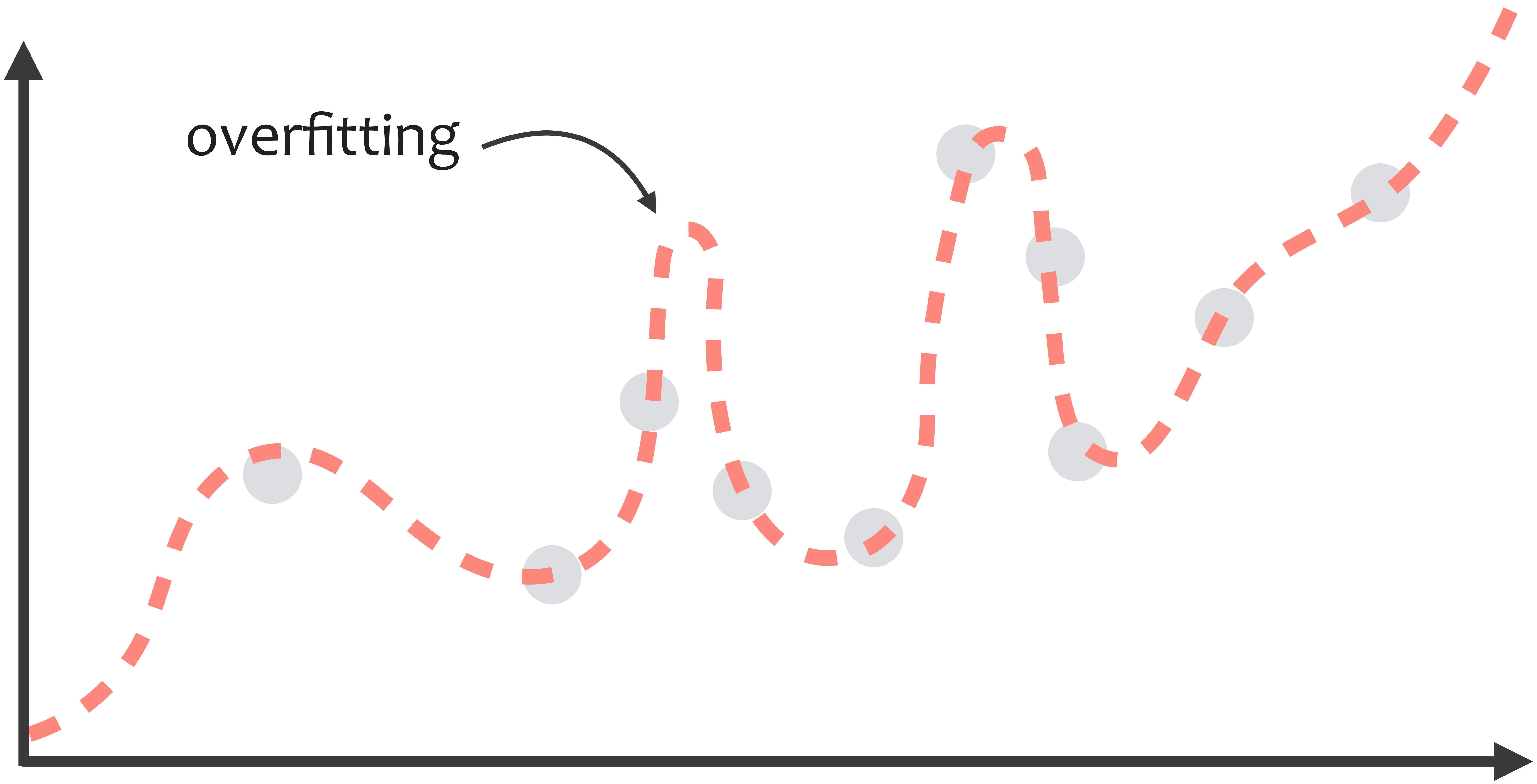
Non-linear Function Approximator



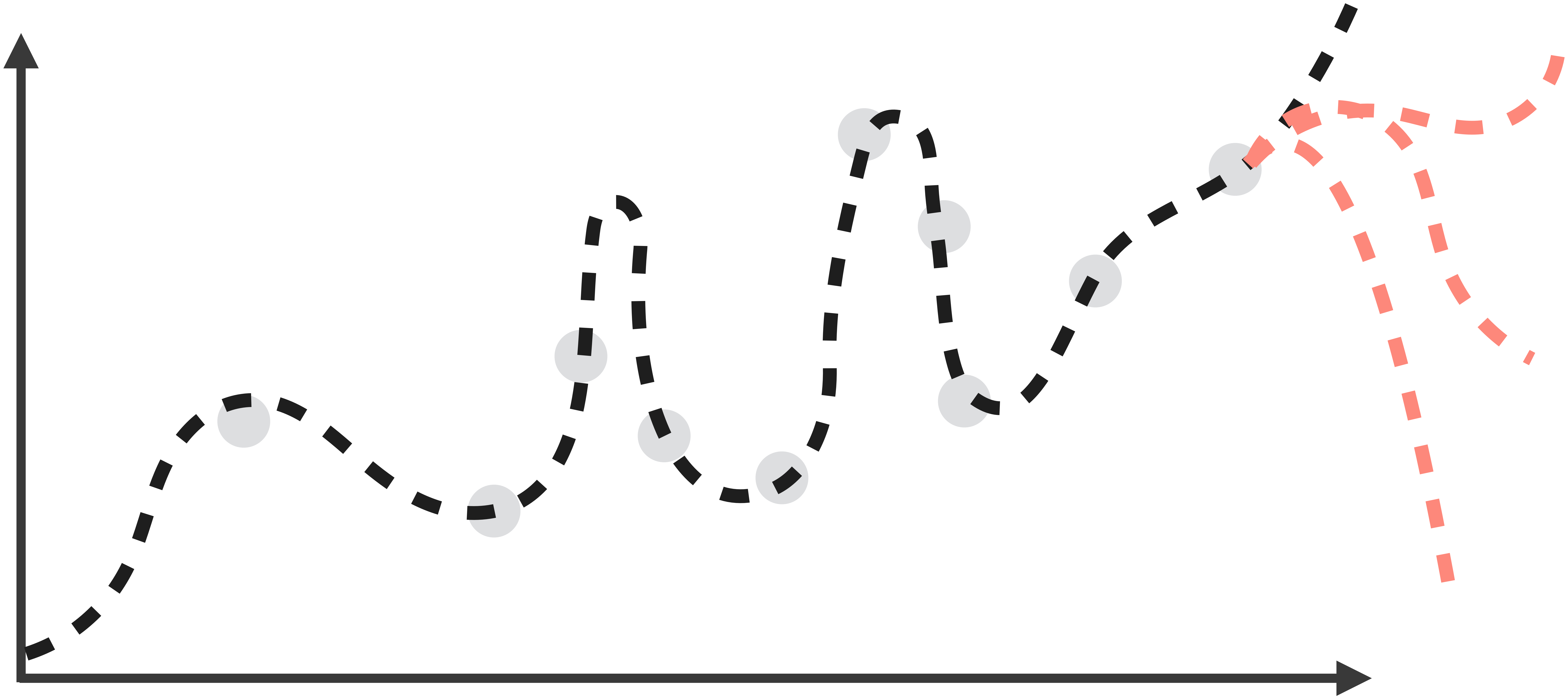
Need a lot of training data



May suffer from overfitting



Difficult to Generalize



Machine learning as a feature extractor

Feature extractor

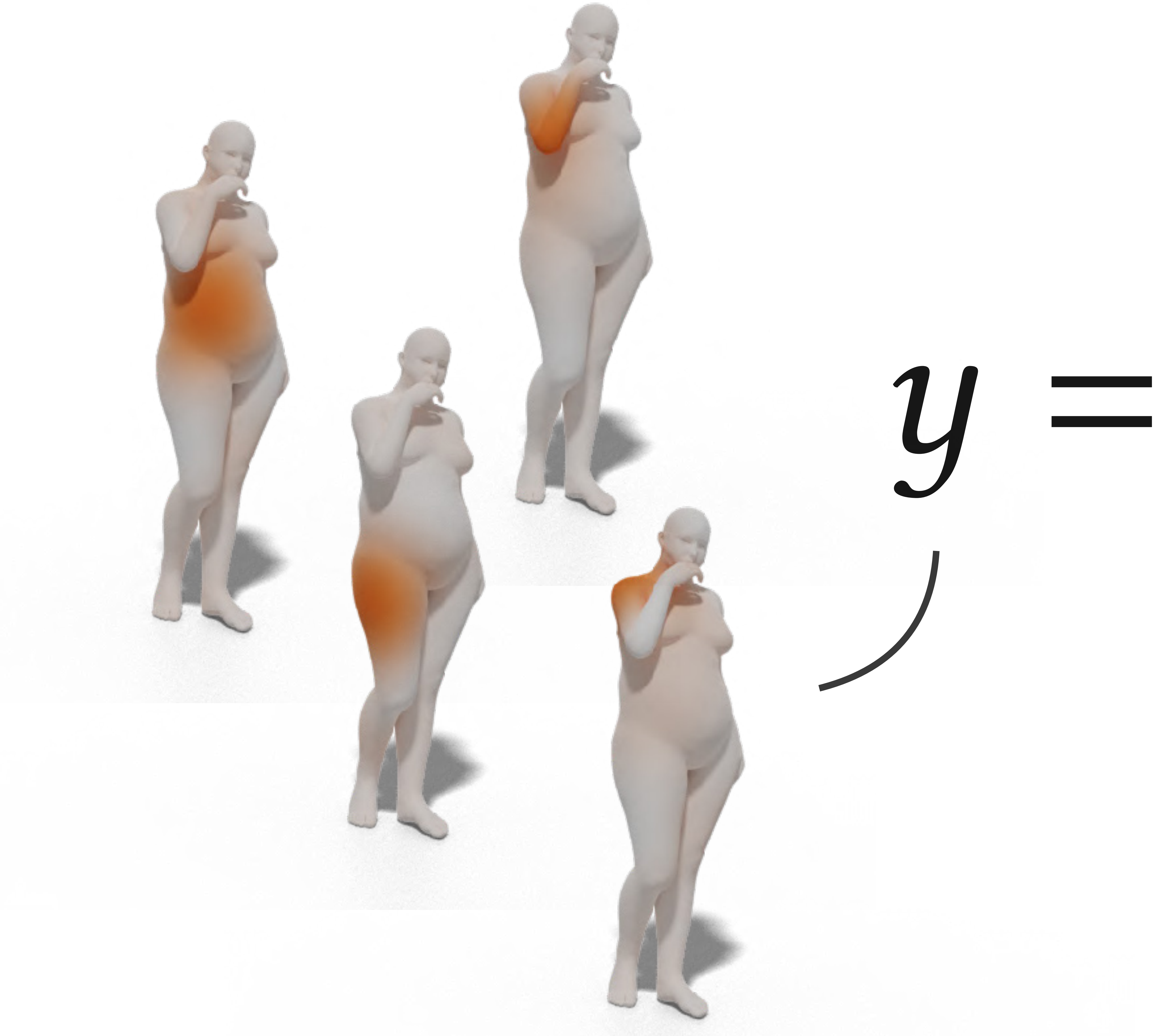
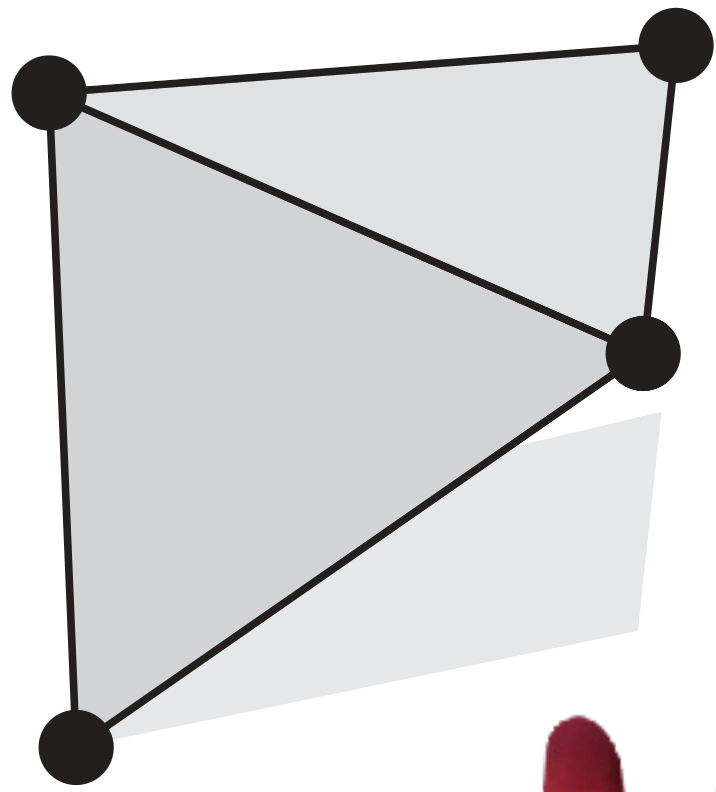
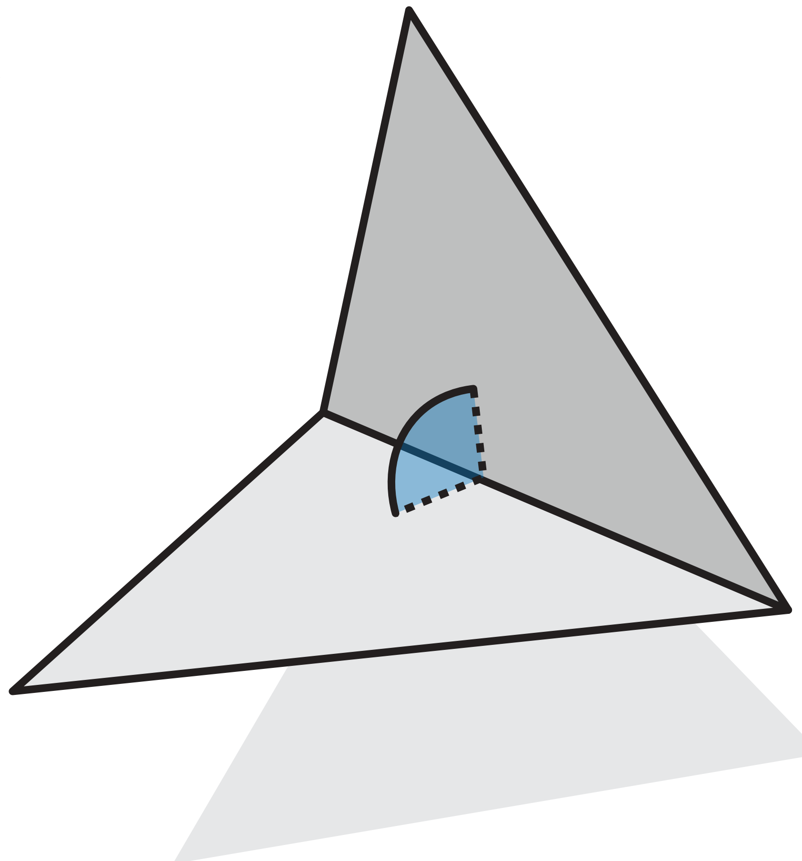


image source: Donati et al. 2020

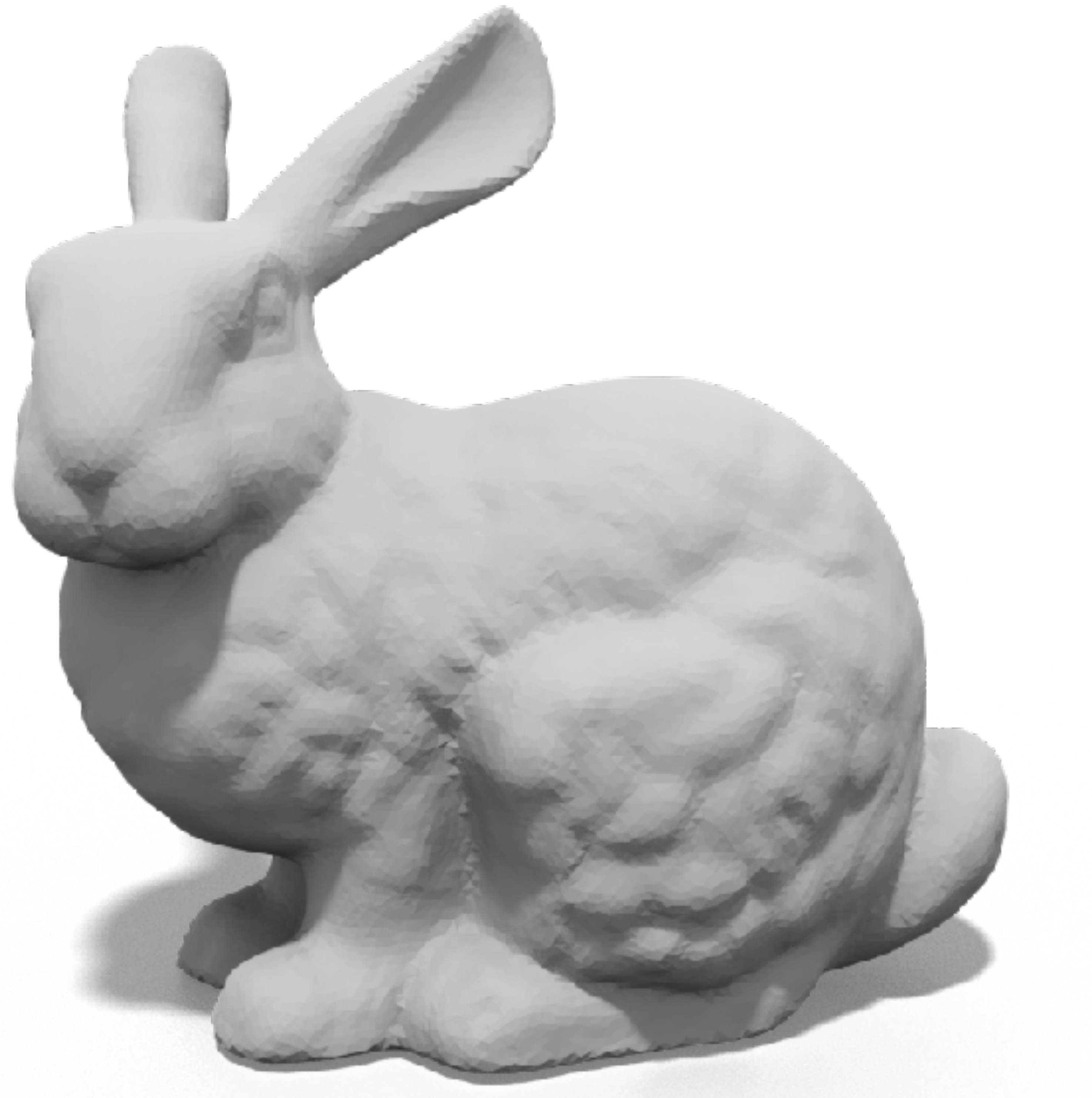
$$y = f_\theta(x)$$





Shape Classification

human



not human

Global Shape Descriptors

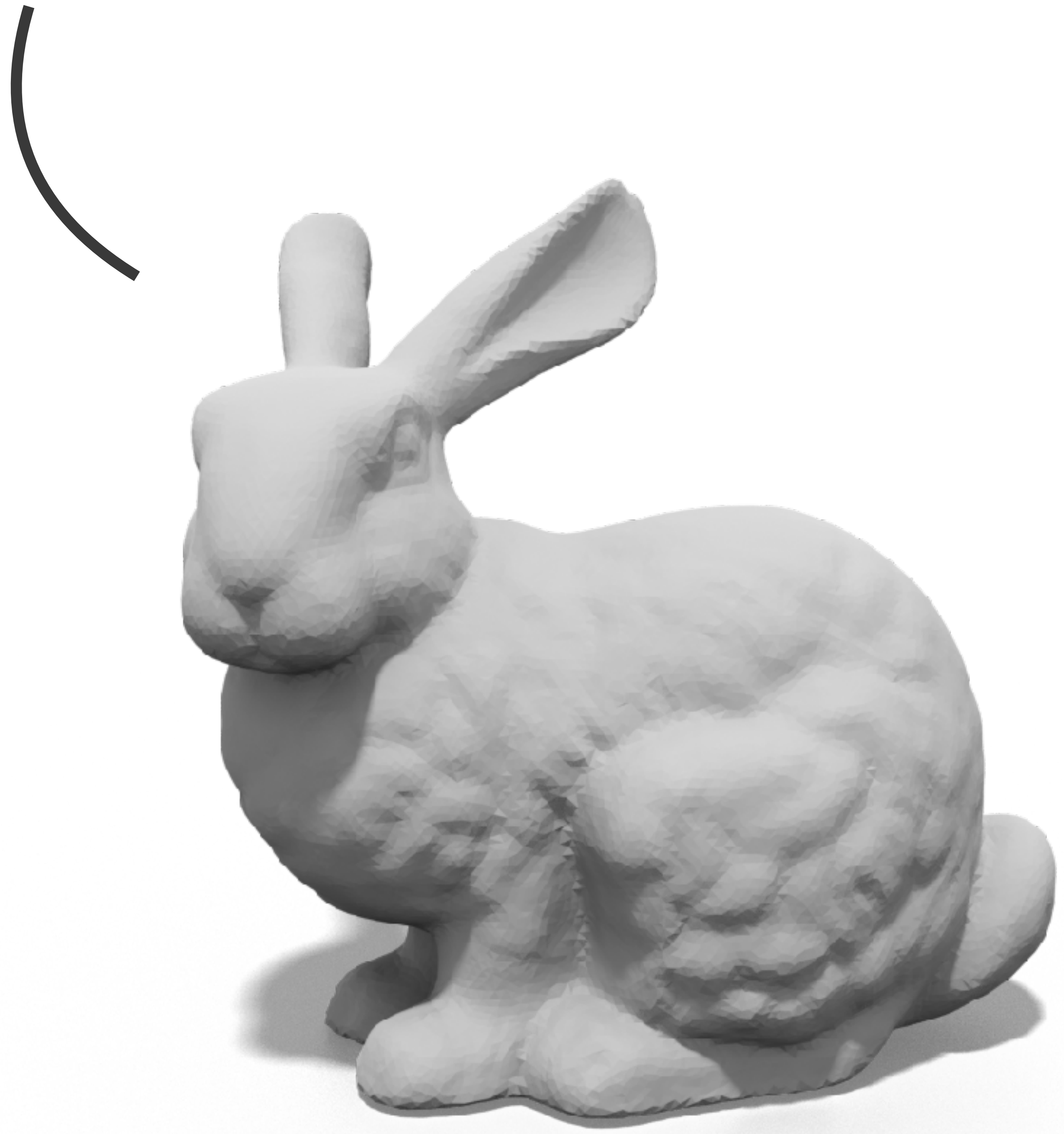
entire shape

$$\rightarrow [a_1, a_2, \cdots, a_n]$$

a fixed dimensional vector

Measure Shape Difference

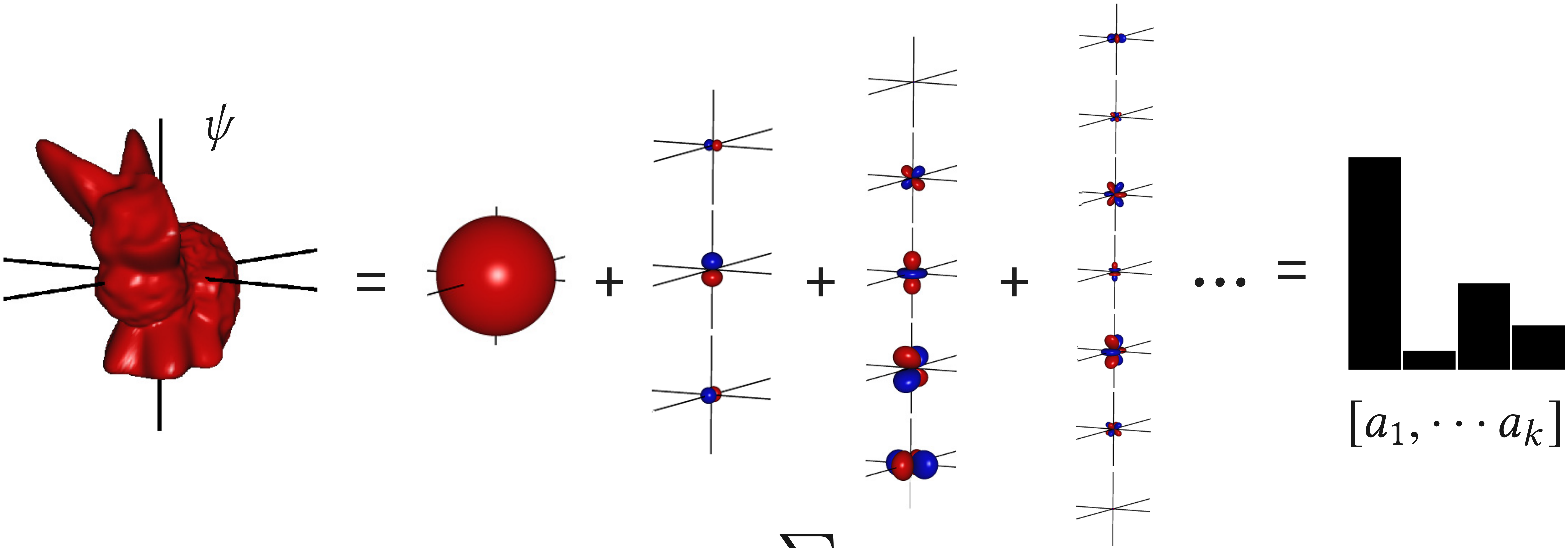
$\| [a_1, a_2, \cdots, a_n] - [b_1, b_2, \cdots, b_n] \| =$ ⬆



Measure Shape Difference

$$\| \, [c_1, c_2, \cdots, c_n] \; - \; [b_1, b_2, \cdots, b_n] \, \| =$$ ↓

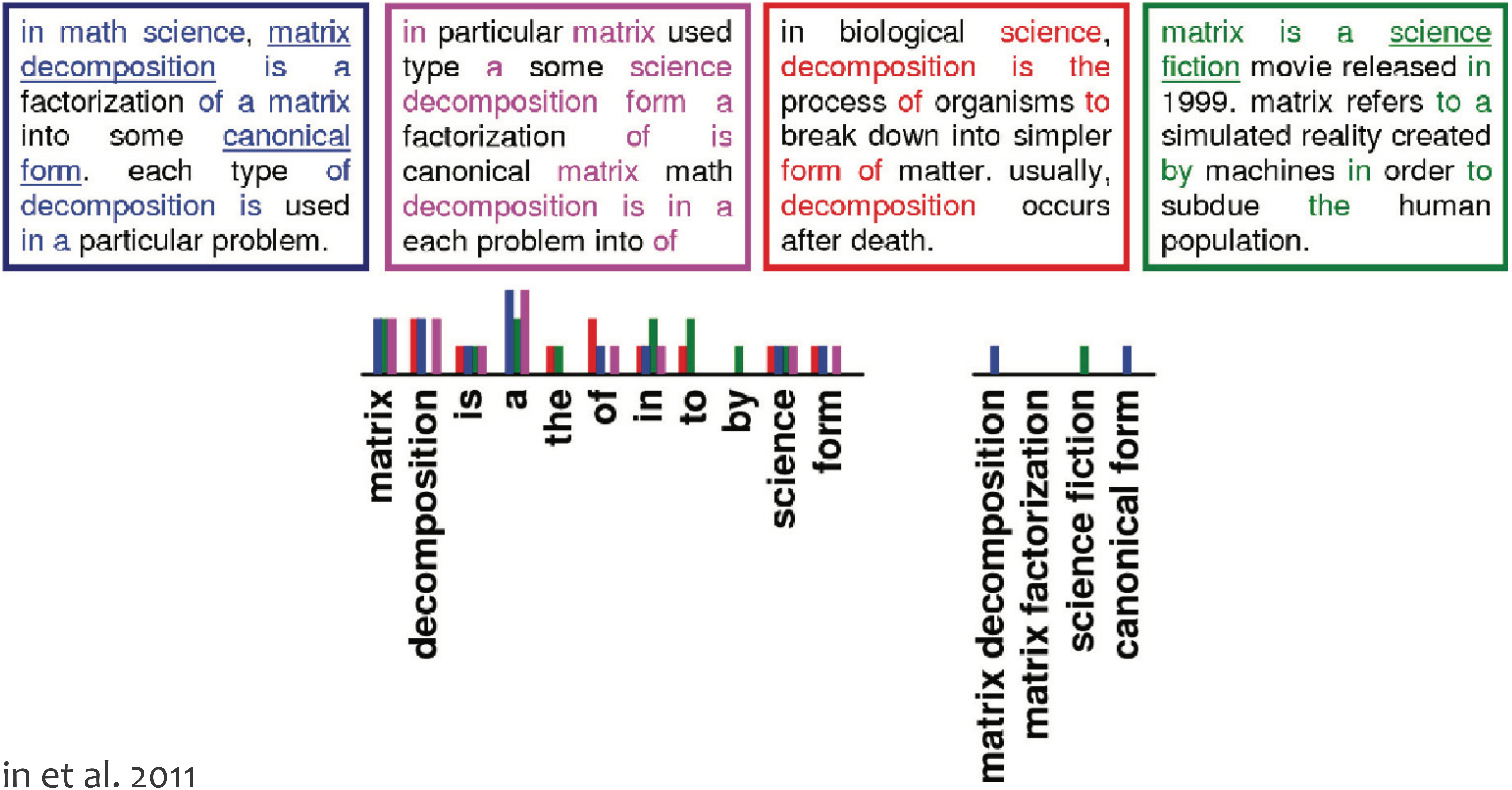
E.g., Spherical Harmonics



$$\psi \approx \sum_{i=1 \cdots k} a_i \phi_i$$

Kazhdan et al. 2003

E.g., Bags of Words



in math science, matrix decomposition is a factorization of a matrix into some canonical form. each type of decomposition is used in a particular problem.

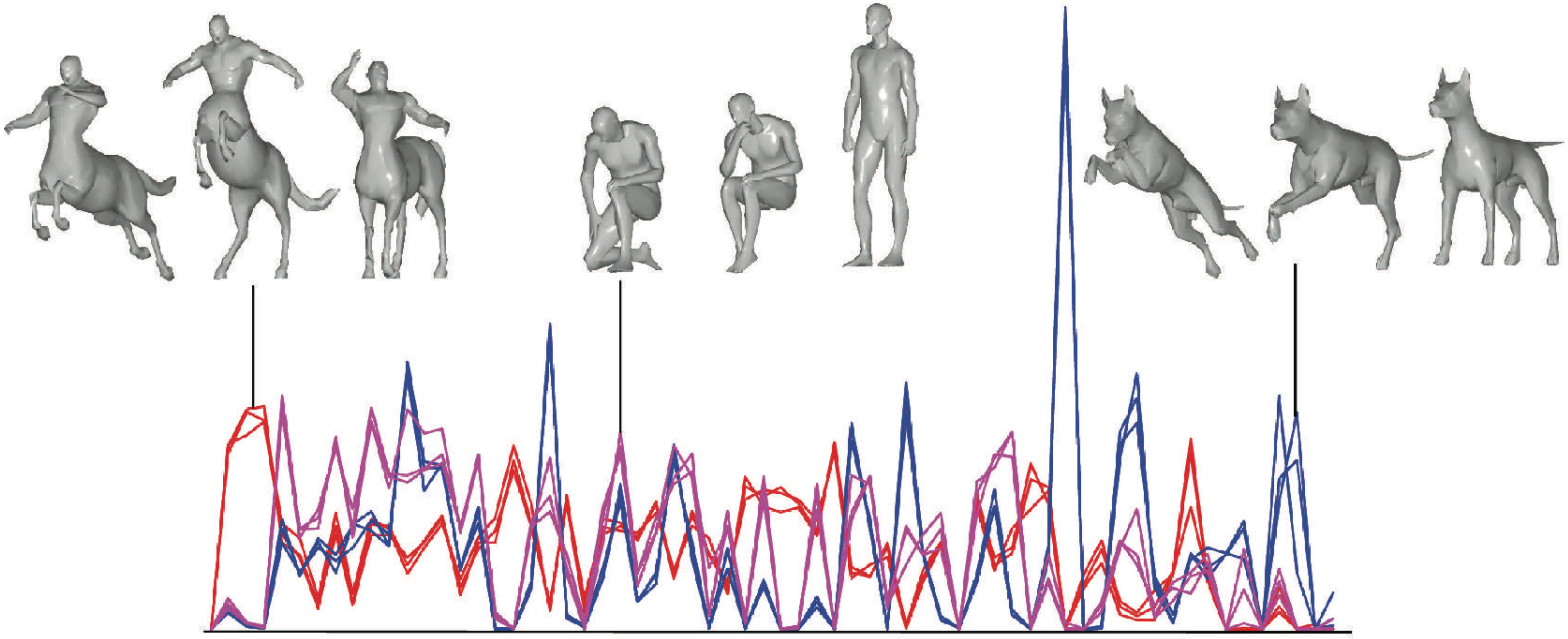
in particular matrix used type a some science decomposition form a factorization of is canonical matrix math decomposition is in a each problem into of

in biological science, decomposition is the process of organisms to break down into simpler form of matter. usually, decomposition occurs after death.

matrix is a science fiction movie released in 1999. matrix refers to a simulated reality created by machines in order to subdue the human population.

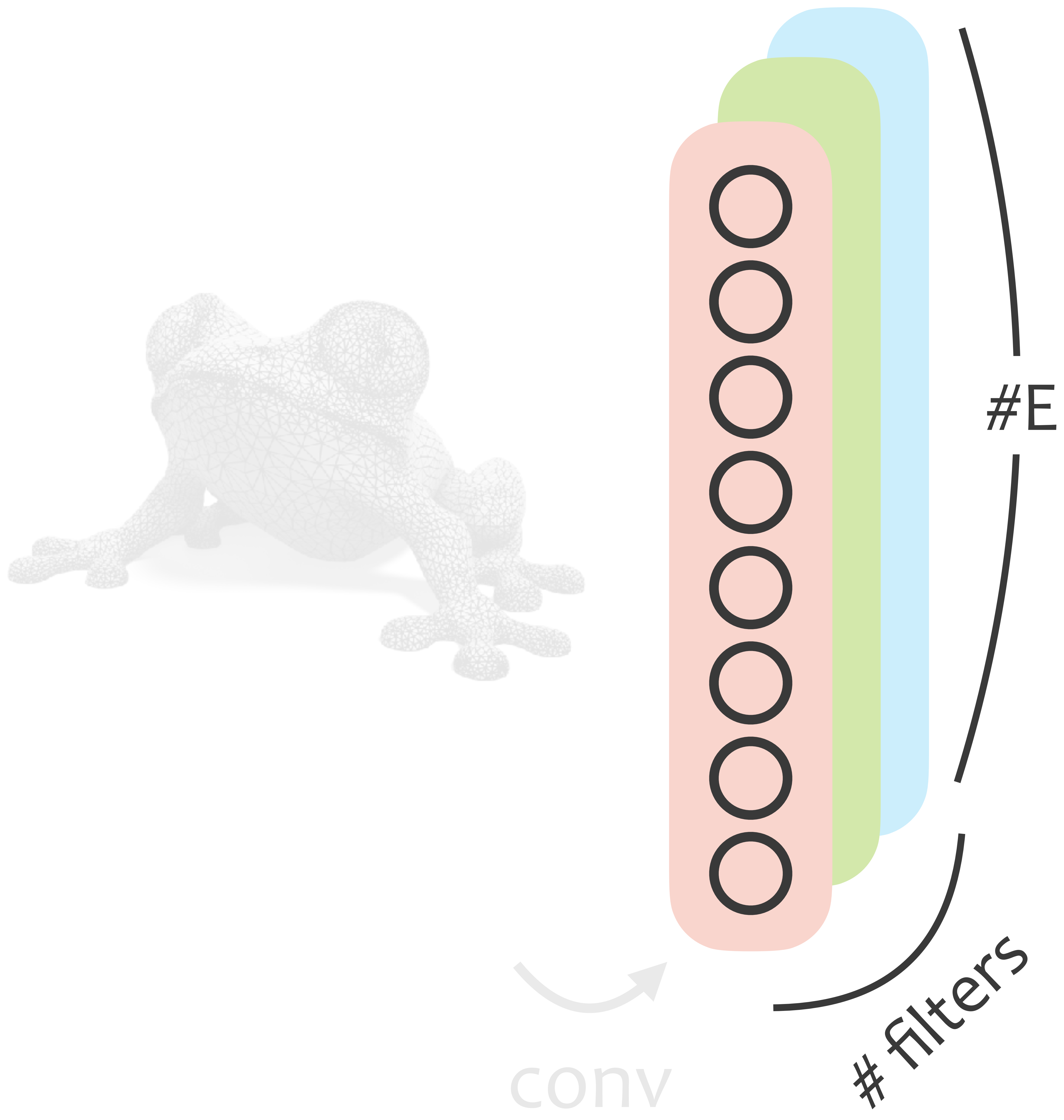
Bronstein et al. 2011

E.g., Bags of Features

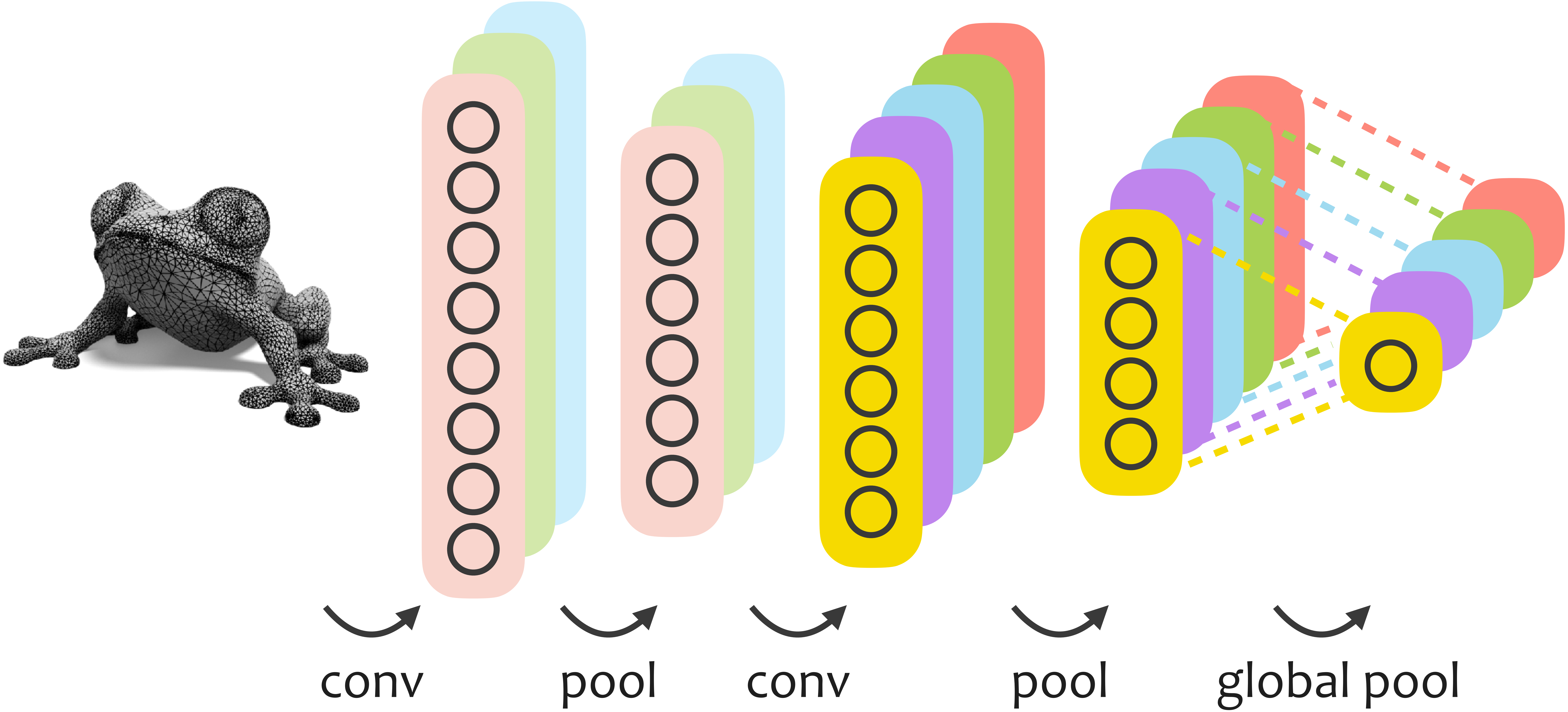


Bronstein et al. 2011

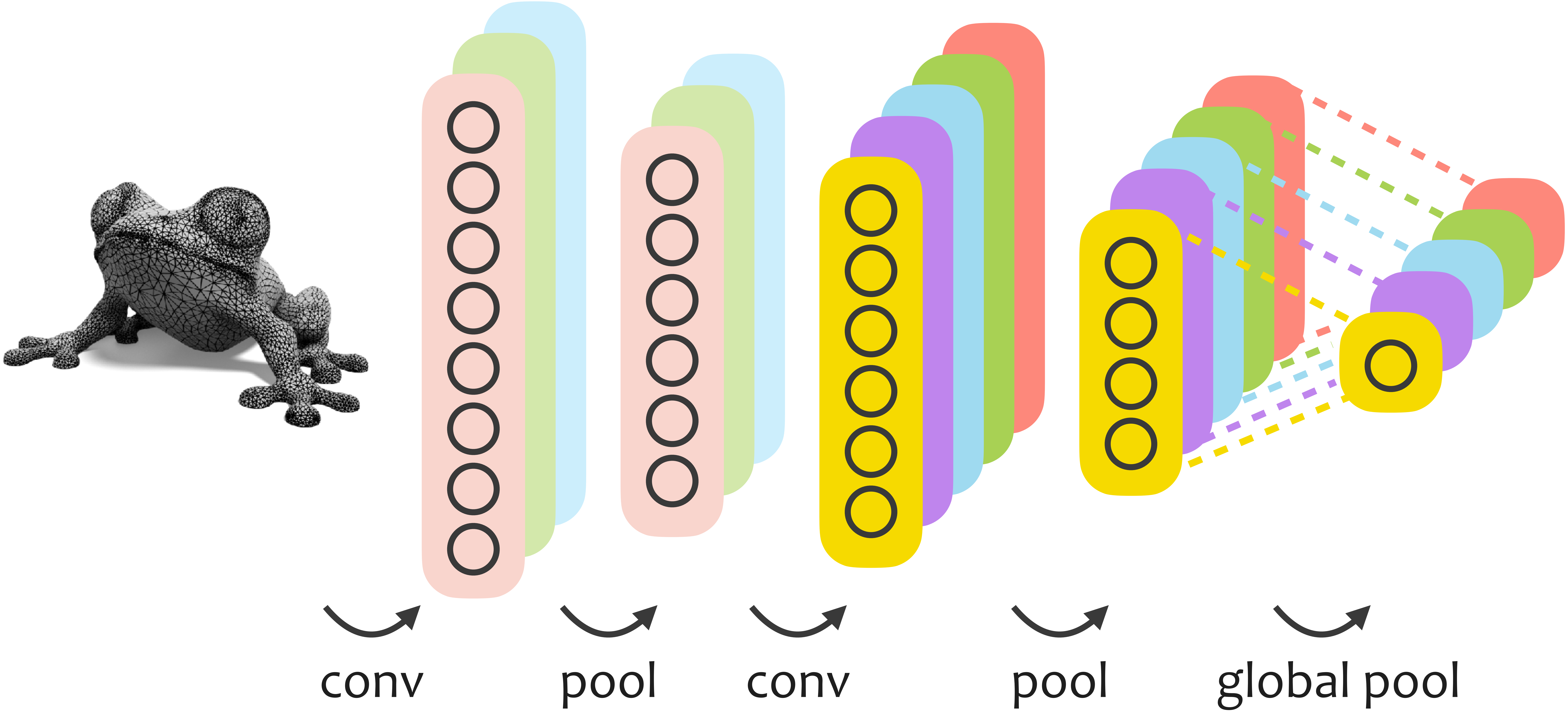
Learned Global Descriptors



Learned Global Descriptors



Learned Global Descriptors



conv

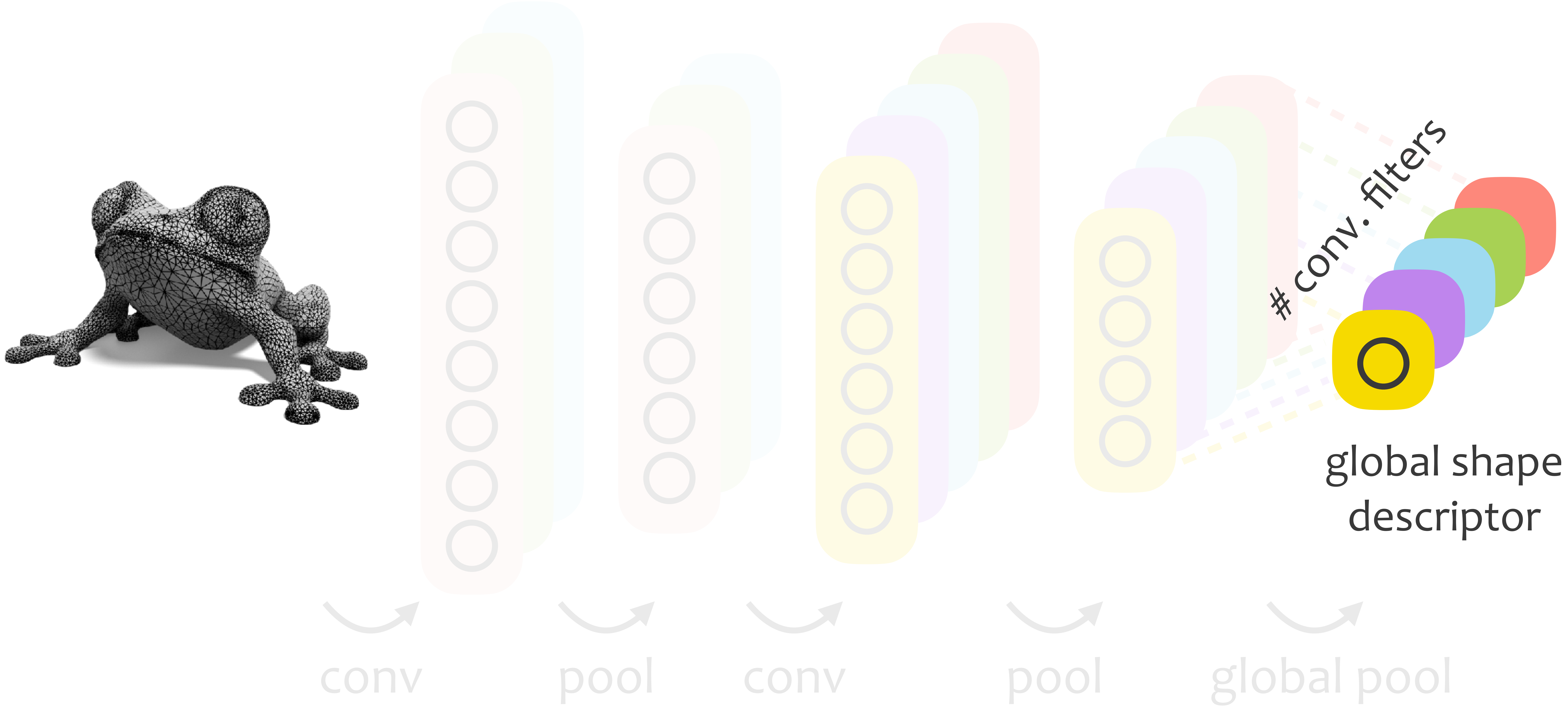
pool

conv

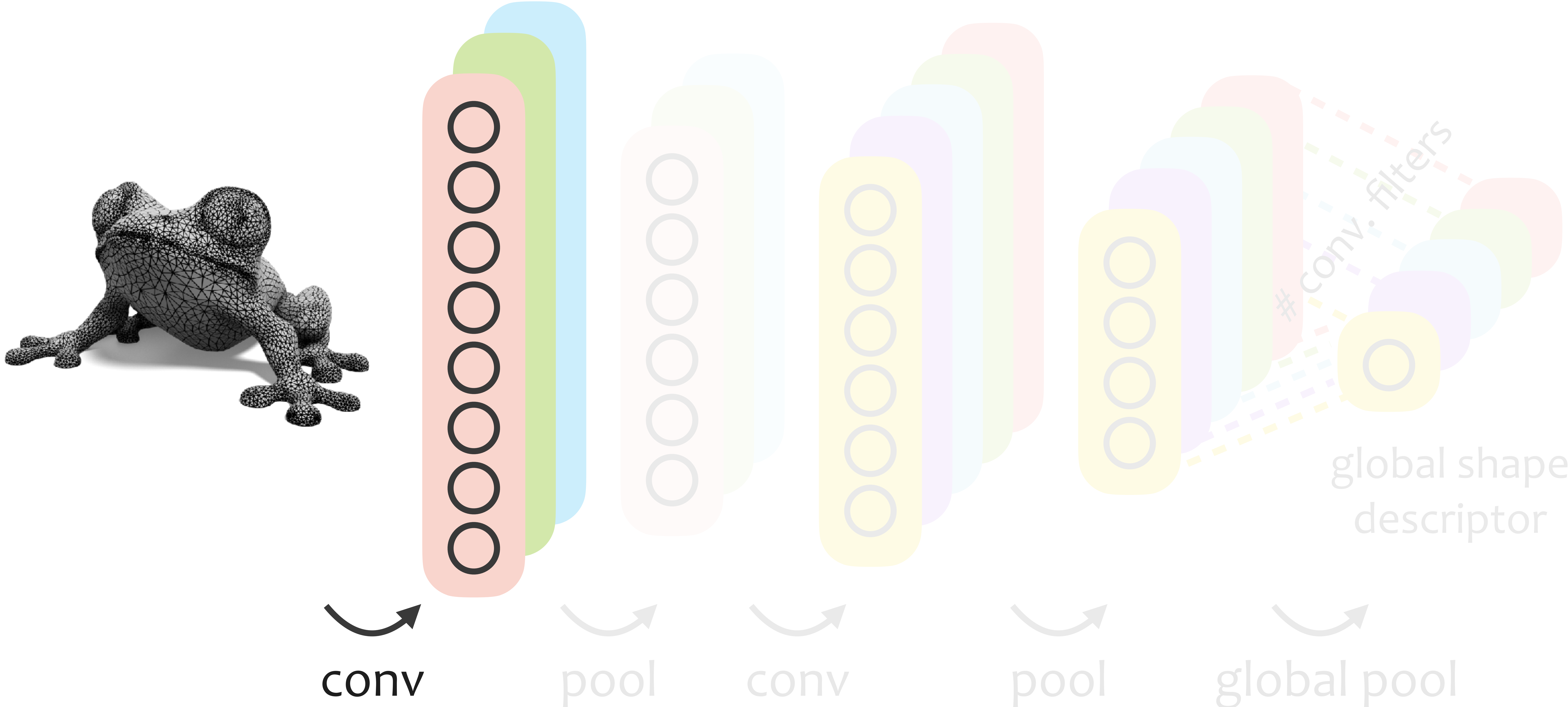
pool

global pool

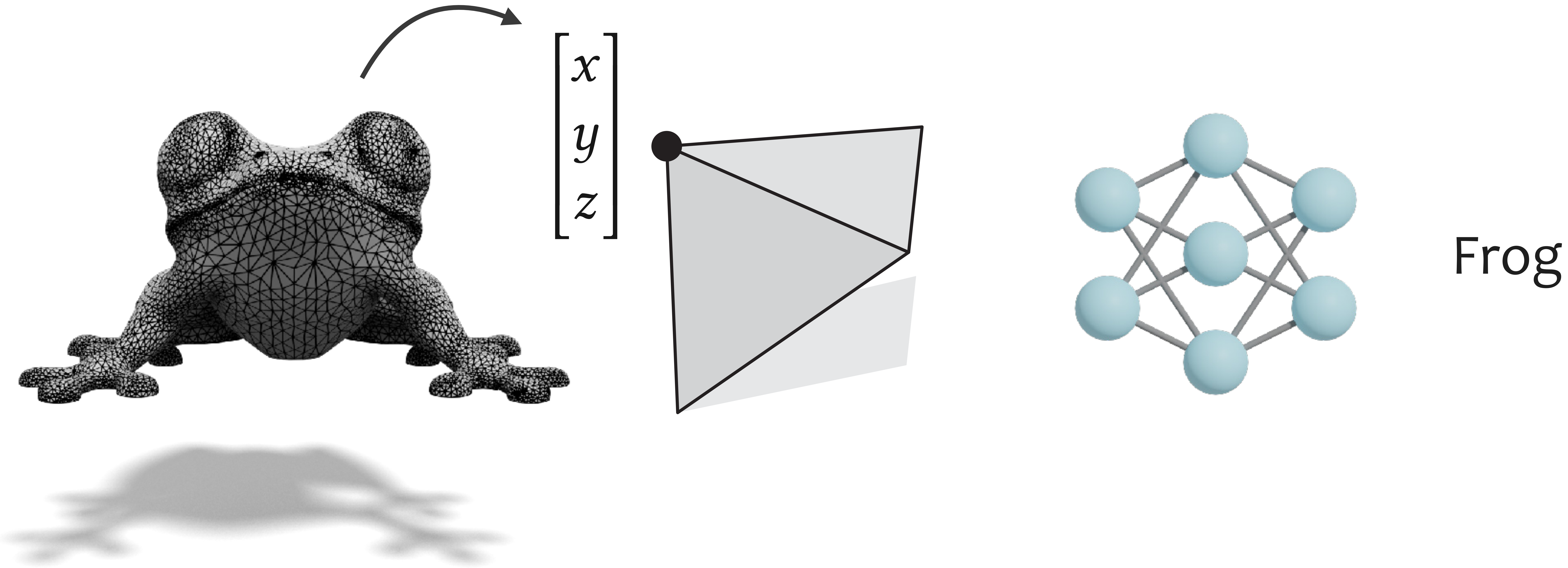
Learned Global Descriptors



Inputs to the network



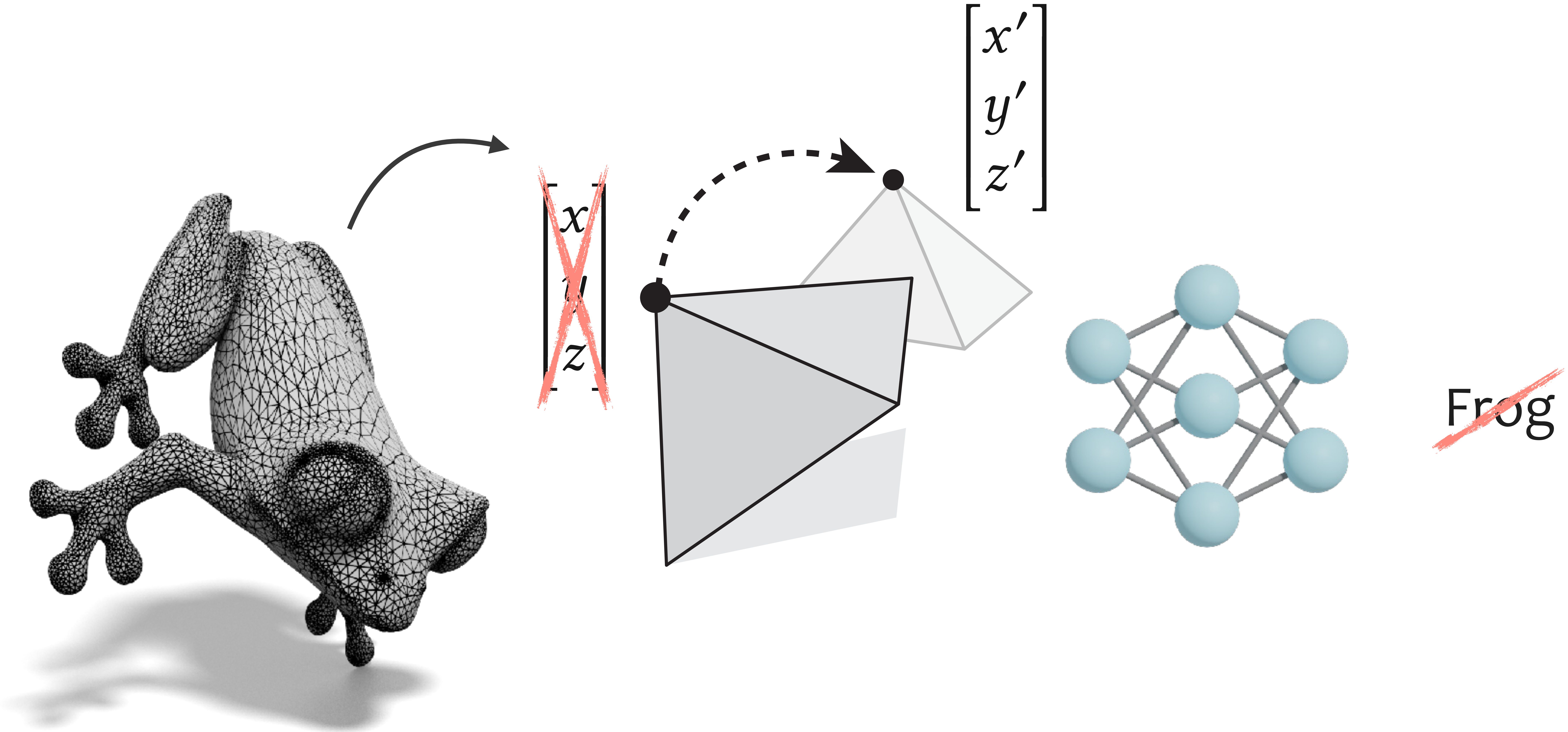
Not Orientation Invariant



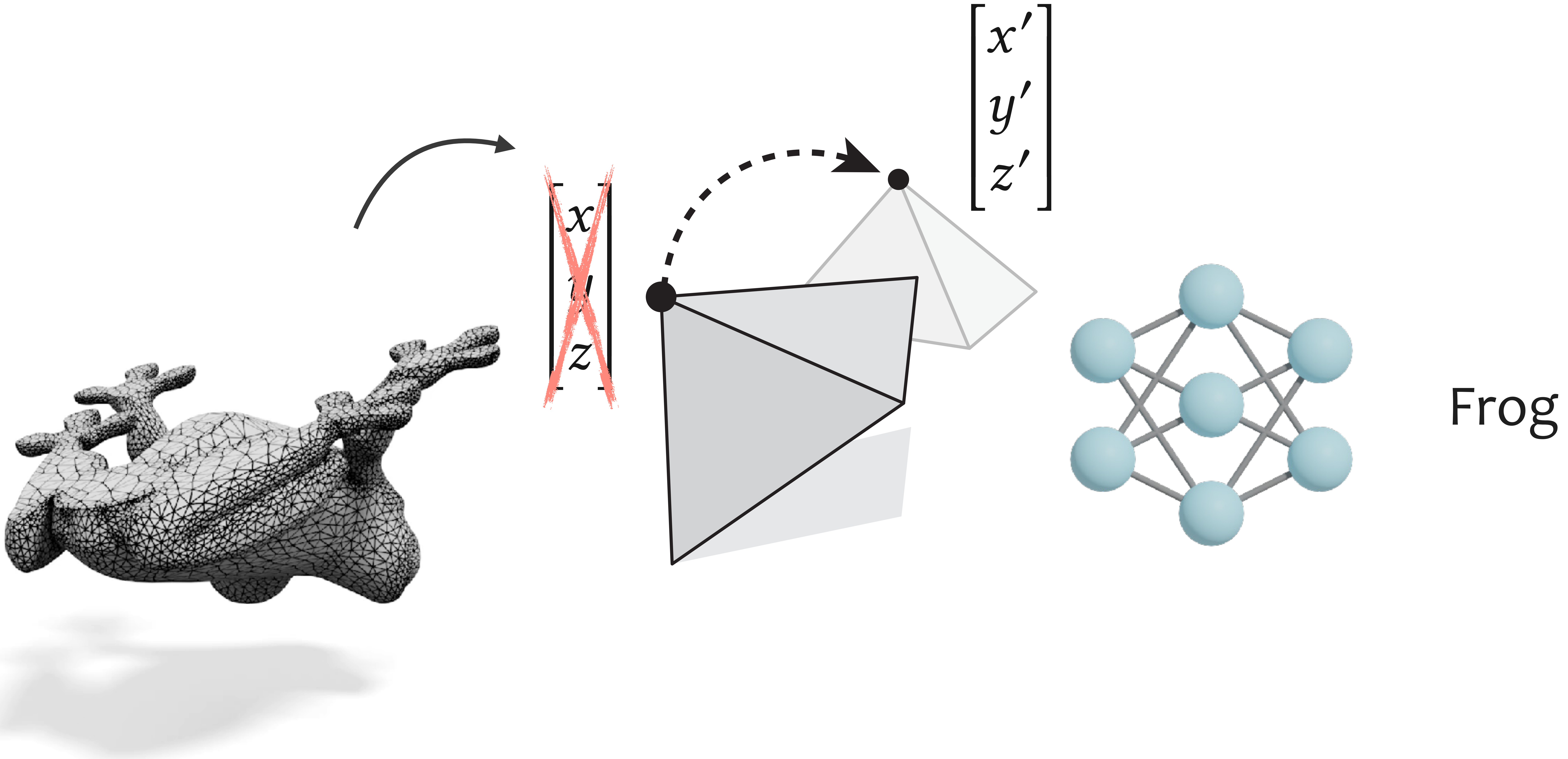
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Frog

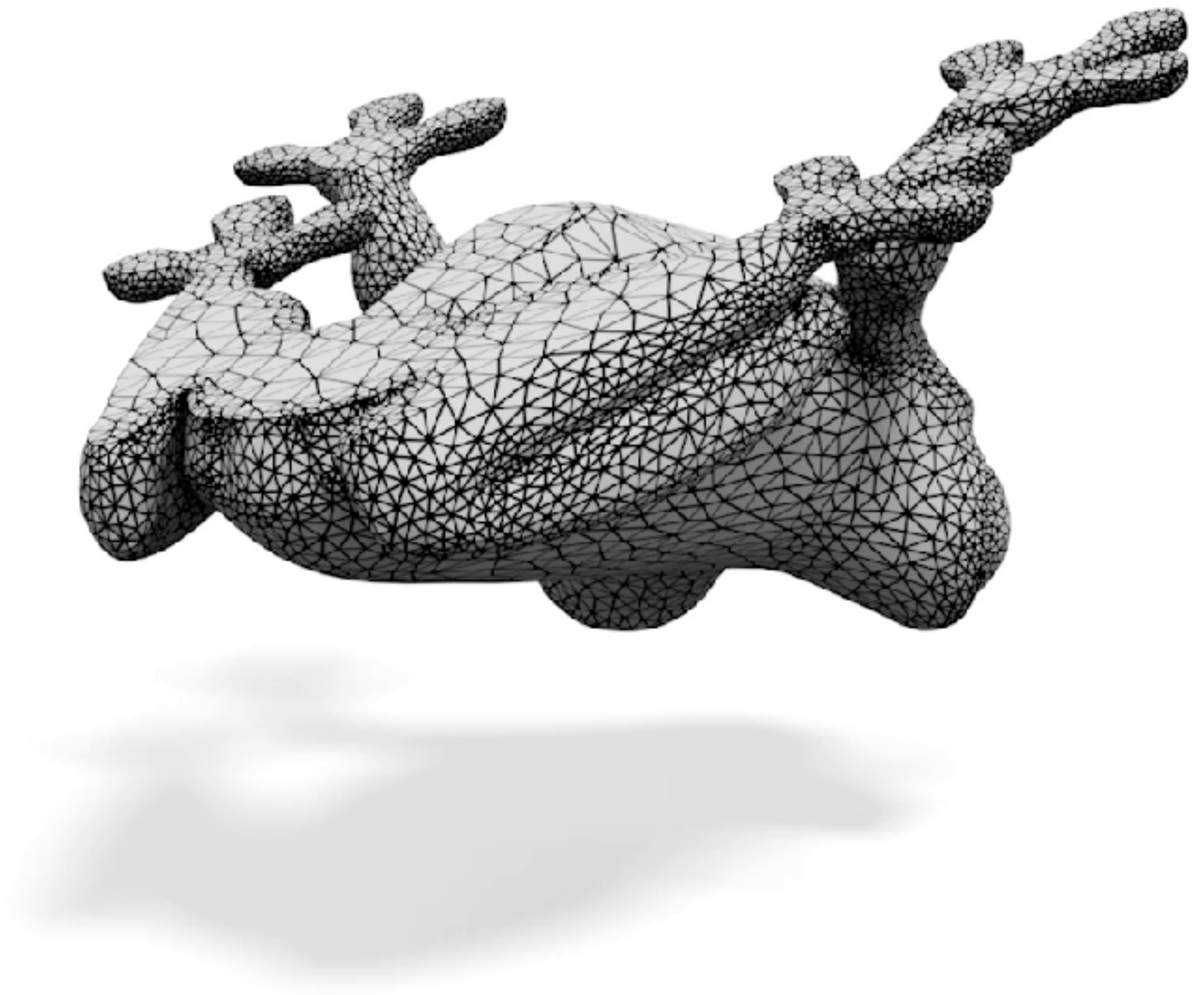
Not Orientation Invariant


$\cancel{\begin{bmatrix}x\\y\\z\end{bmatrix}}$
$\begin{bmatrix}x'\\y'\\z'\end{bmatrix}$
~~Frog~~

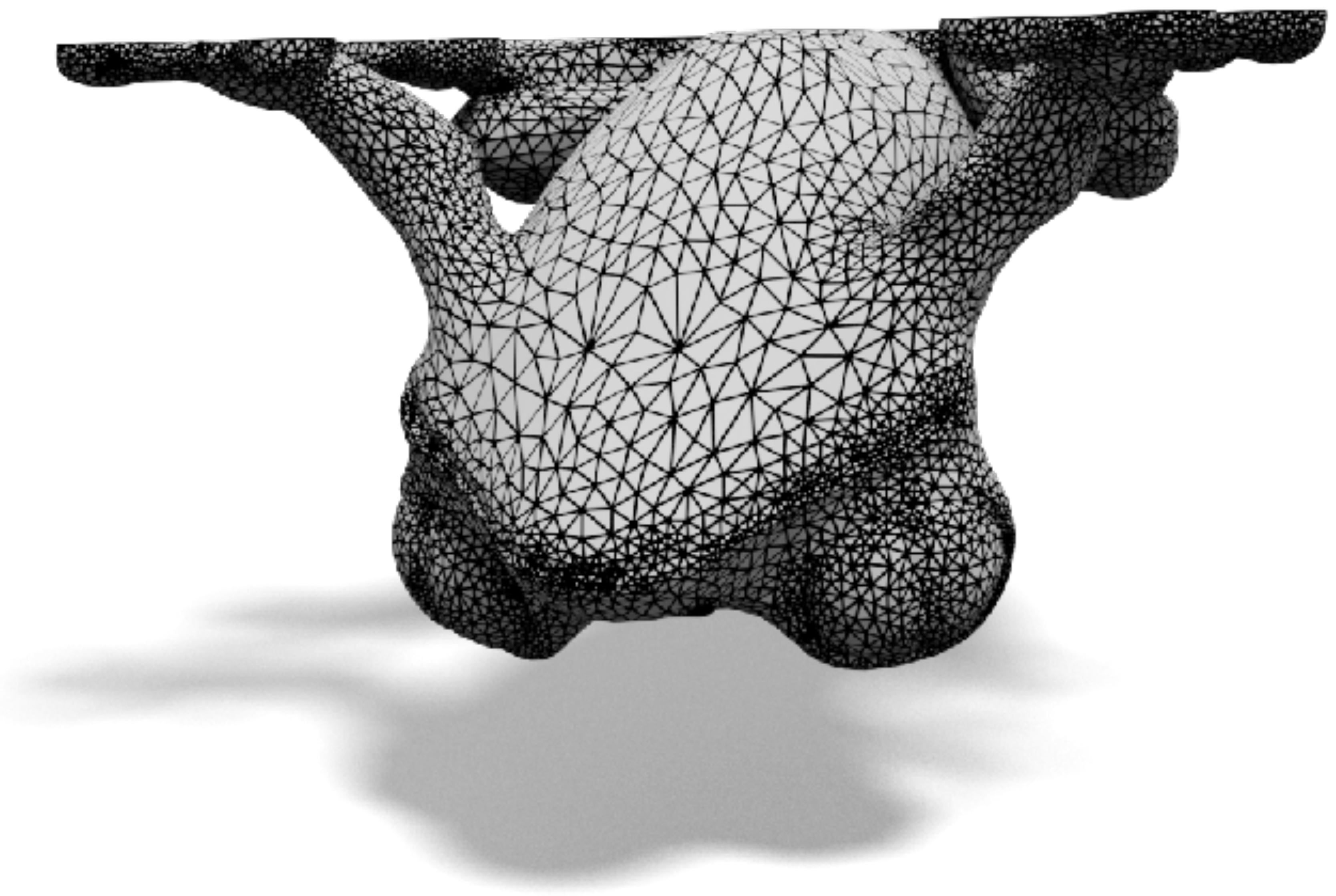
One Solution: Data Augmentation


$\cancel{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}$
$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$
Frog

It helps, but ...

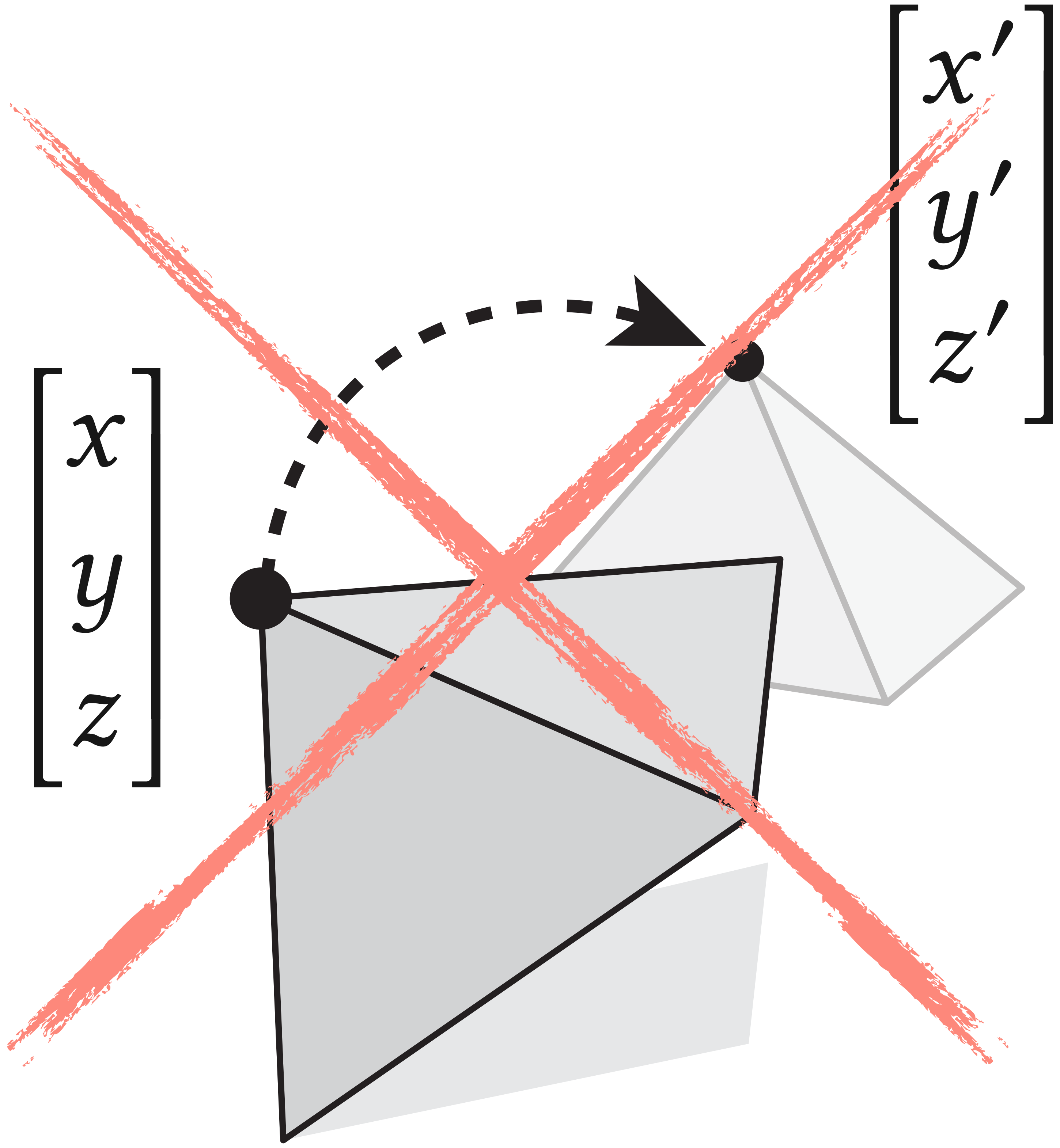


longer to train

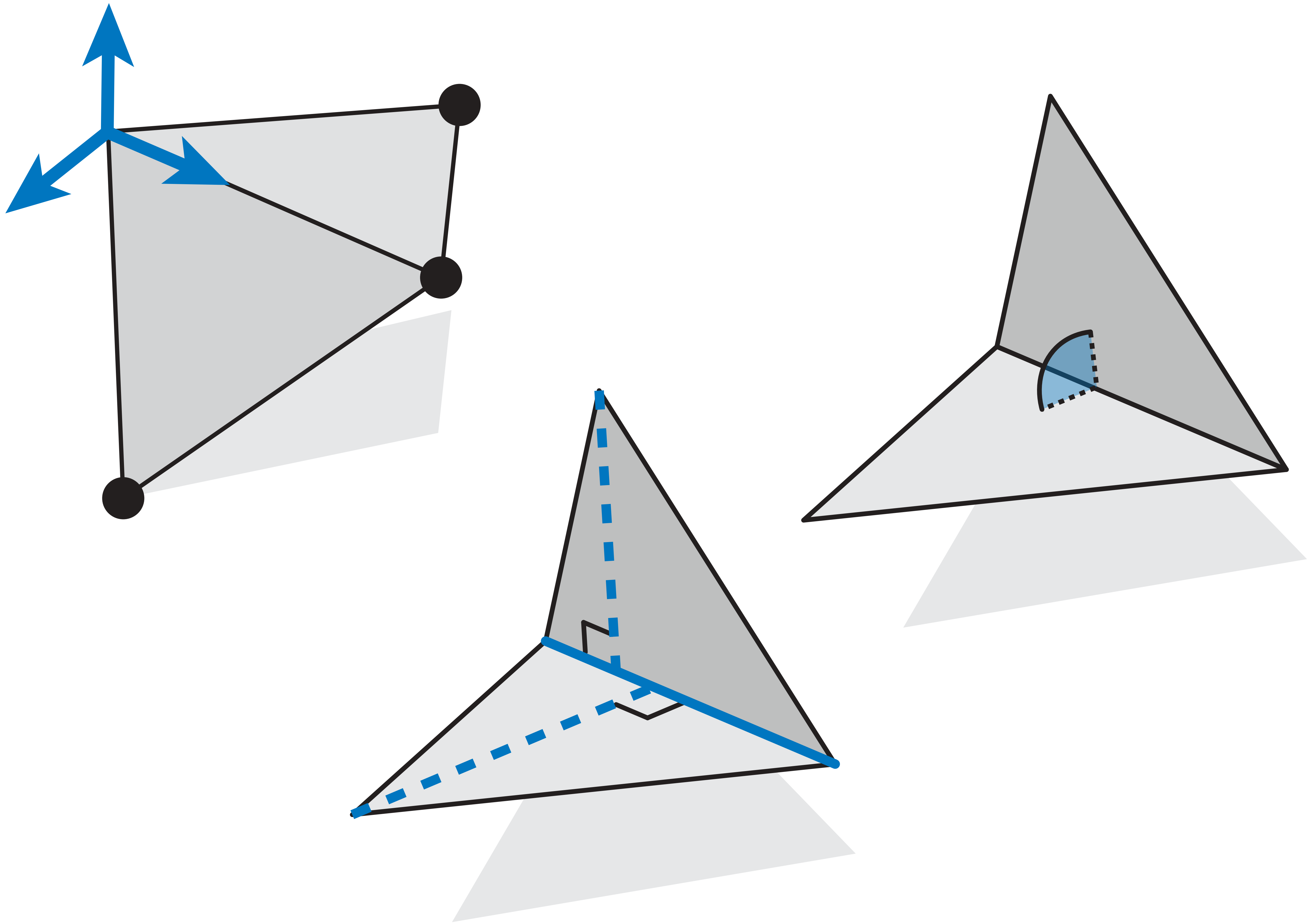


Adversarial attack

Leverage mesh structure

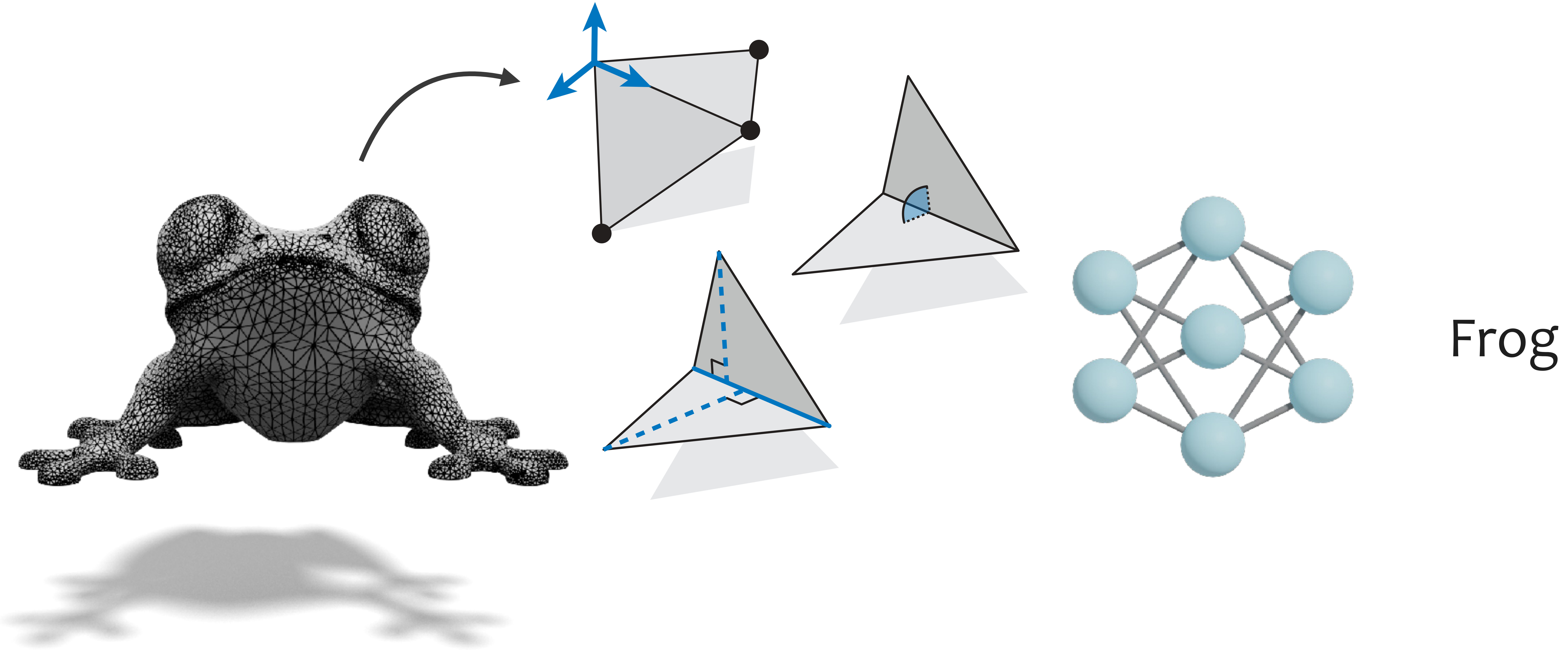

$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$
$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$

not invariant quantities

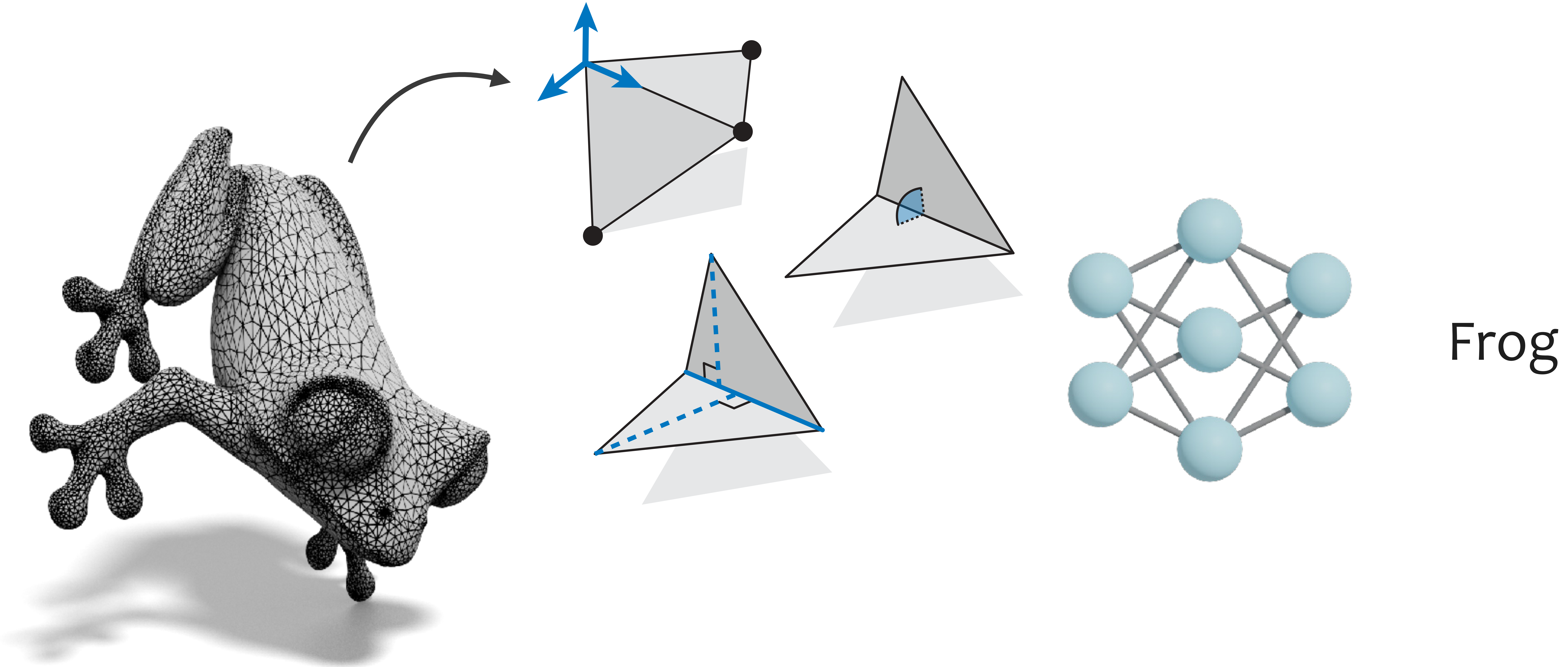


rotation invariant quantities

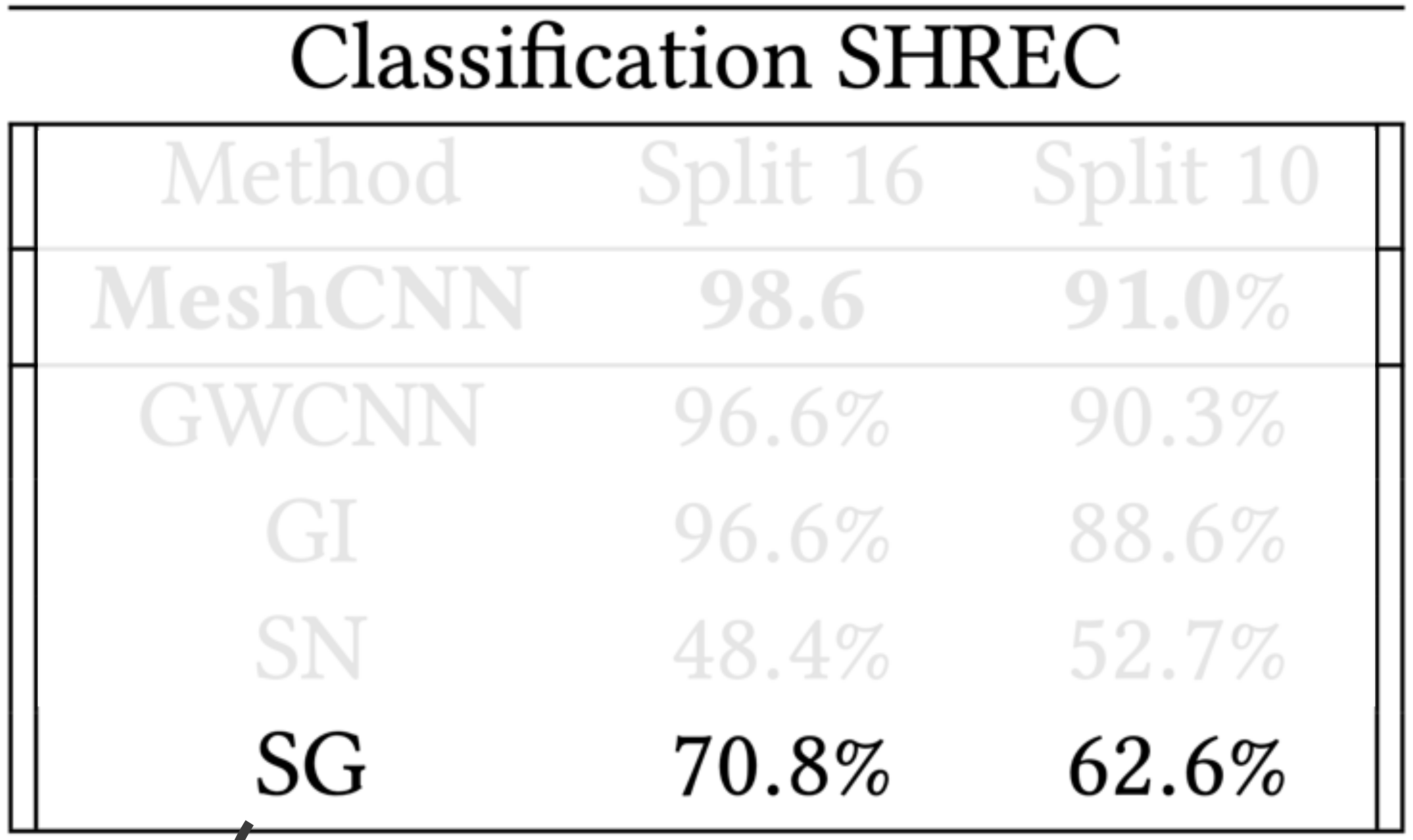
Orientation Invariant



Orientation Invariant



Push the limits

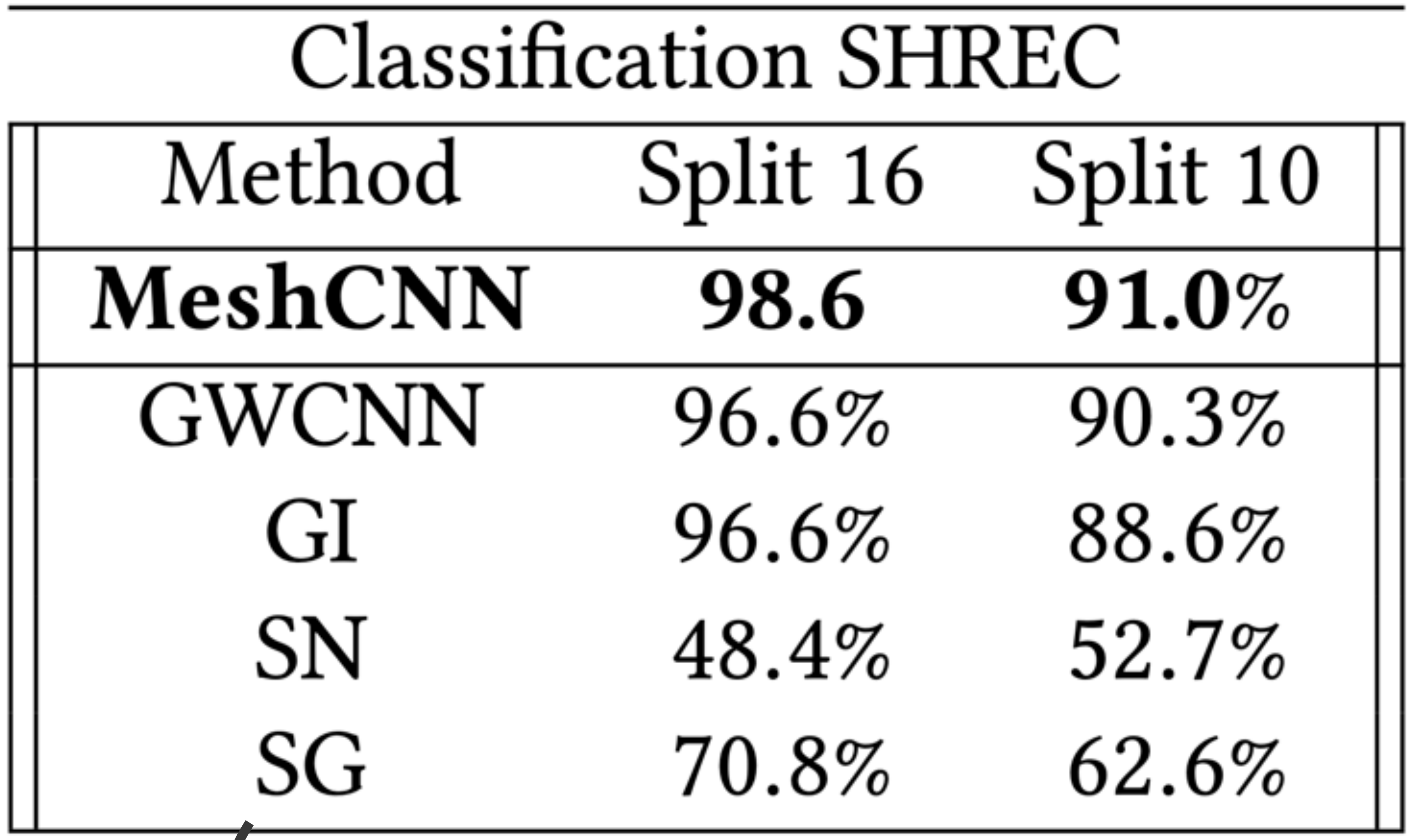


Classification SHREC

Method	Split 16	Split 10
MeshCNN	**98.6**	**91.0%**
GWCNN	96.6%	90.3%
GI	96.6%	88.6%
SN	48.4%	52.7%
SG	70.8%	62.6%

Classic method
[Bronstein et al. 2011]

Push the limits

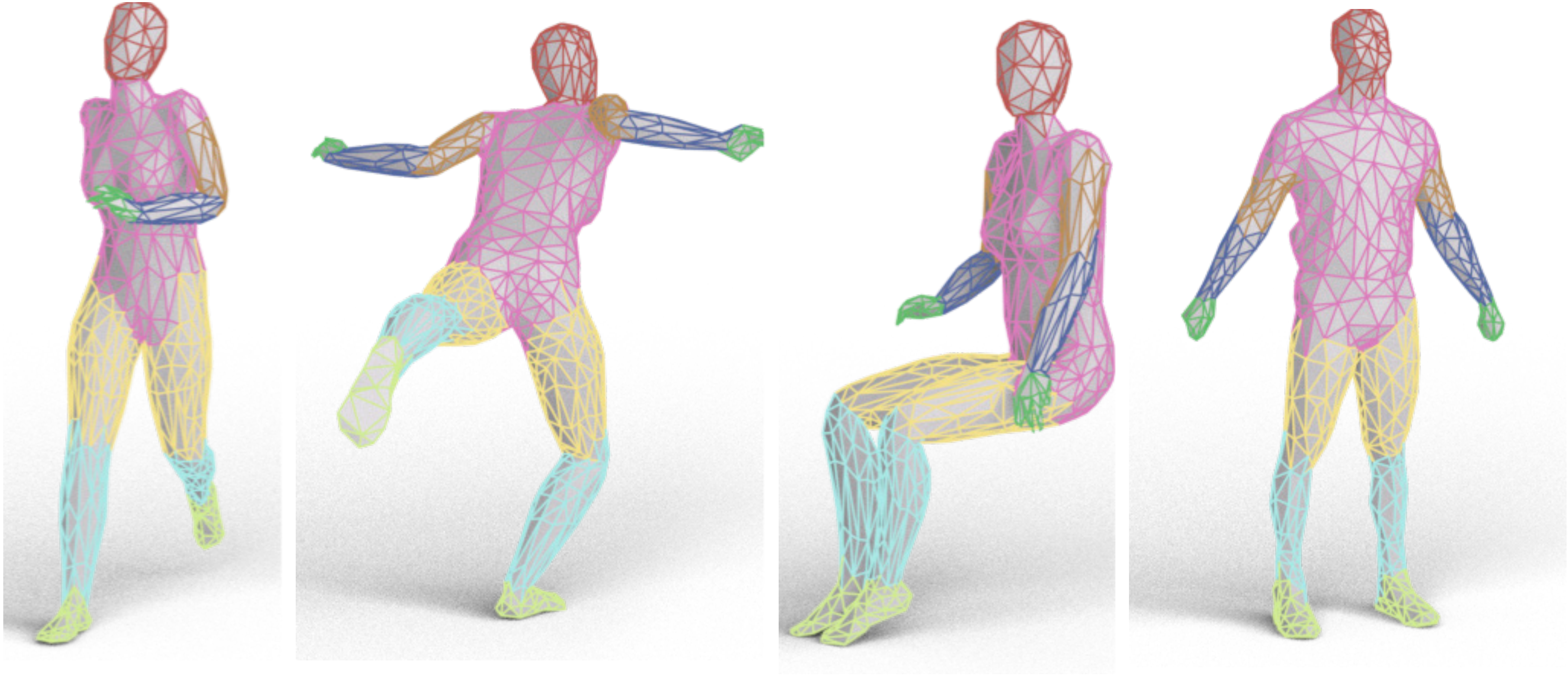


Classification SHREC

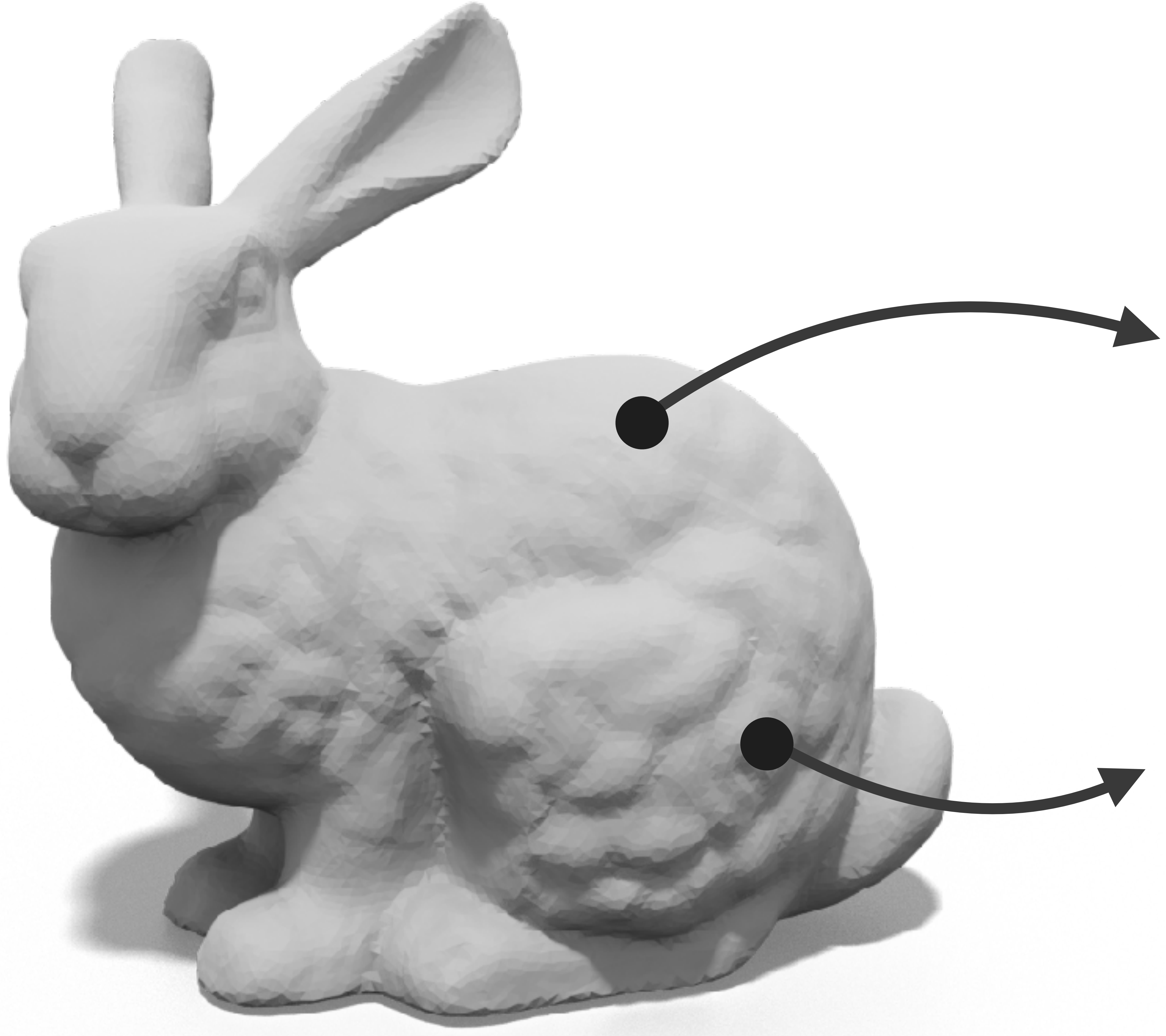
Method	Split 16	Split 10
MeshCNN	**98.6**	**91.0**%
GWCNN	96.6%	90.3%
GI	96.6%	88.6%
SN	48.4%	52.7%
SG	70.8%	62.6%

Classic method
[Bronstein et al. 2011]

Shape Segmentation



Local Shape Descriptors

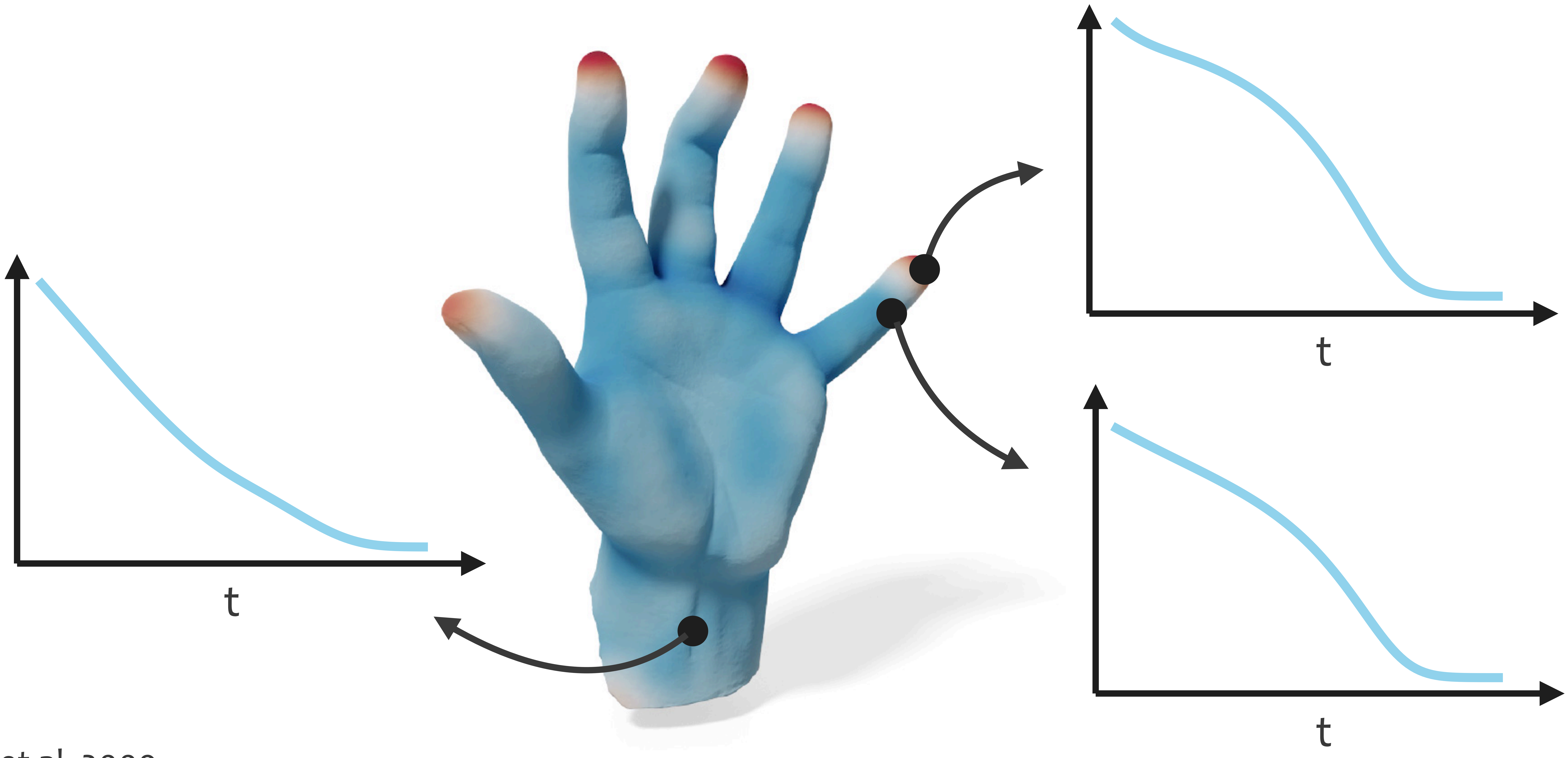


$$[a_1, a_2, \cdots, a_n]$$

$$[b_1, b_2, \cdots, b_n]$$

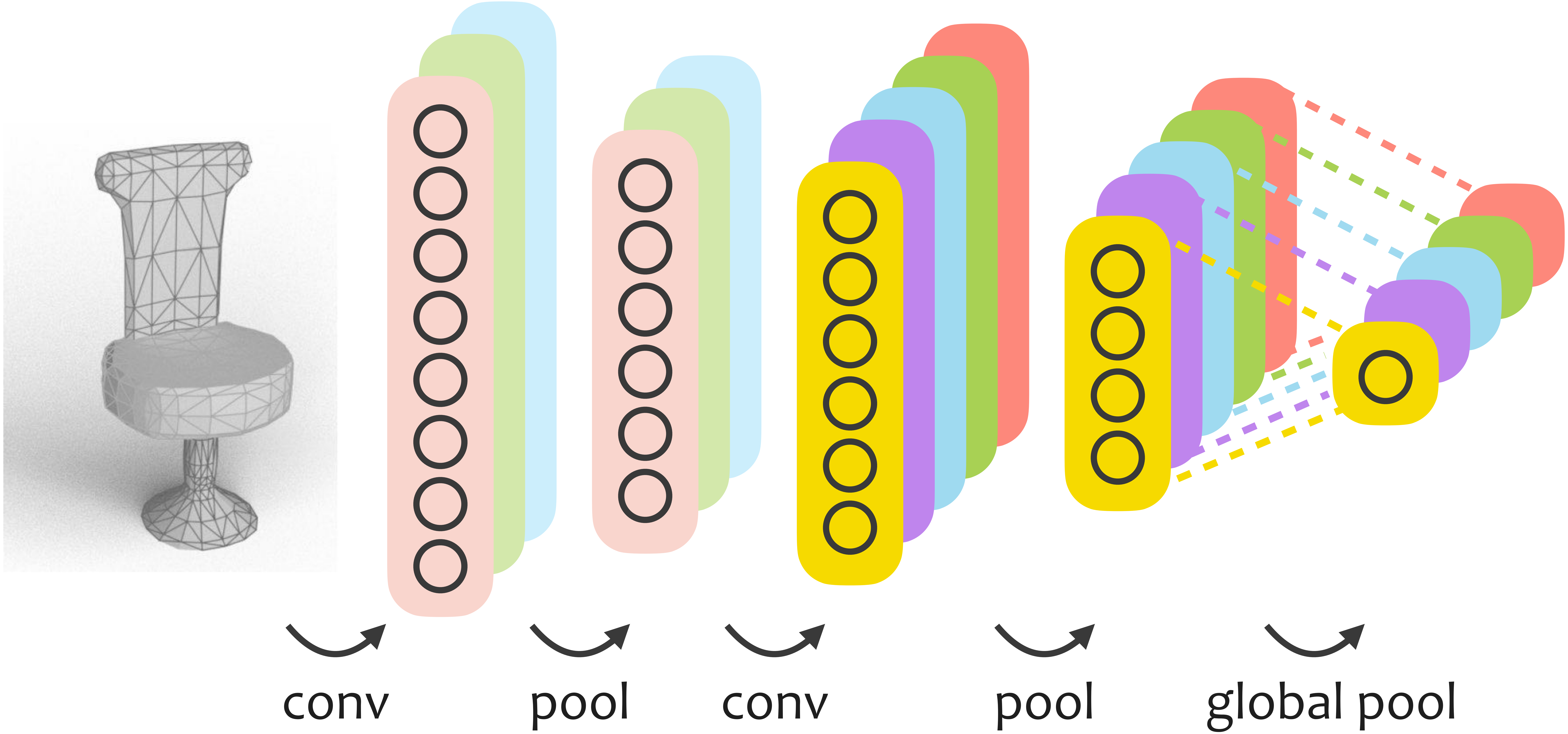
a fixed dimensional vector
per element

E.g., Heat Kernel Signature

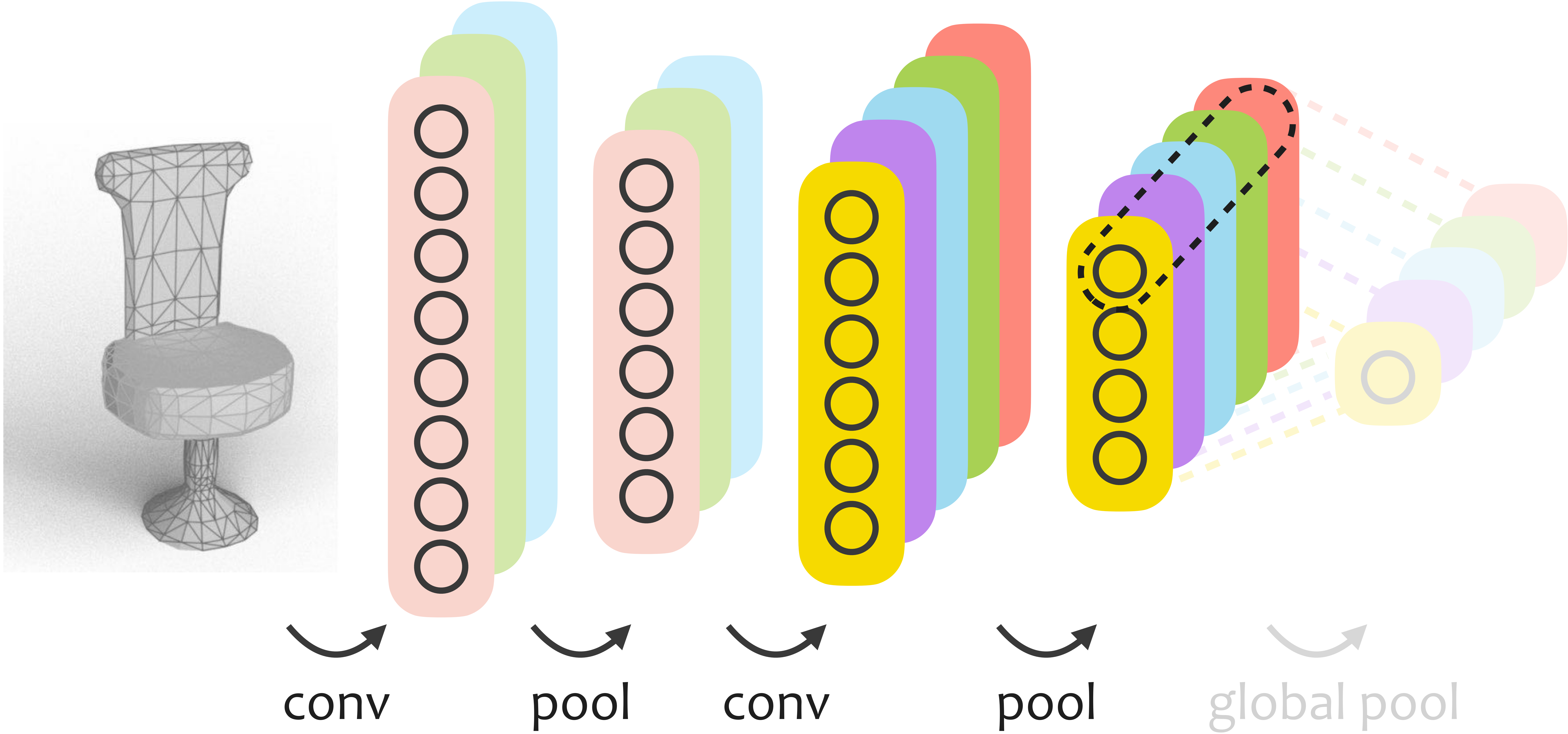


Sun et al. 2009

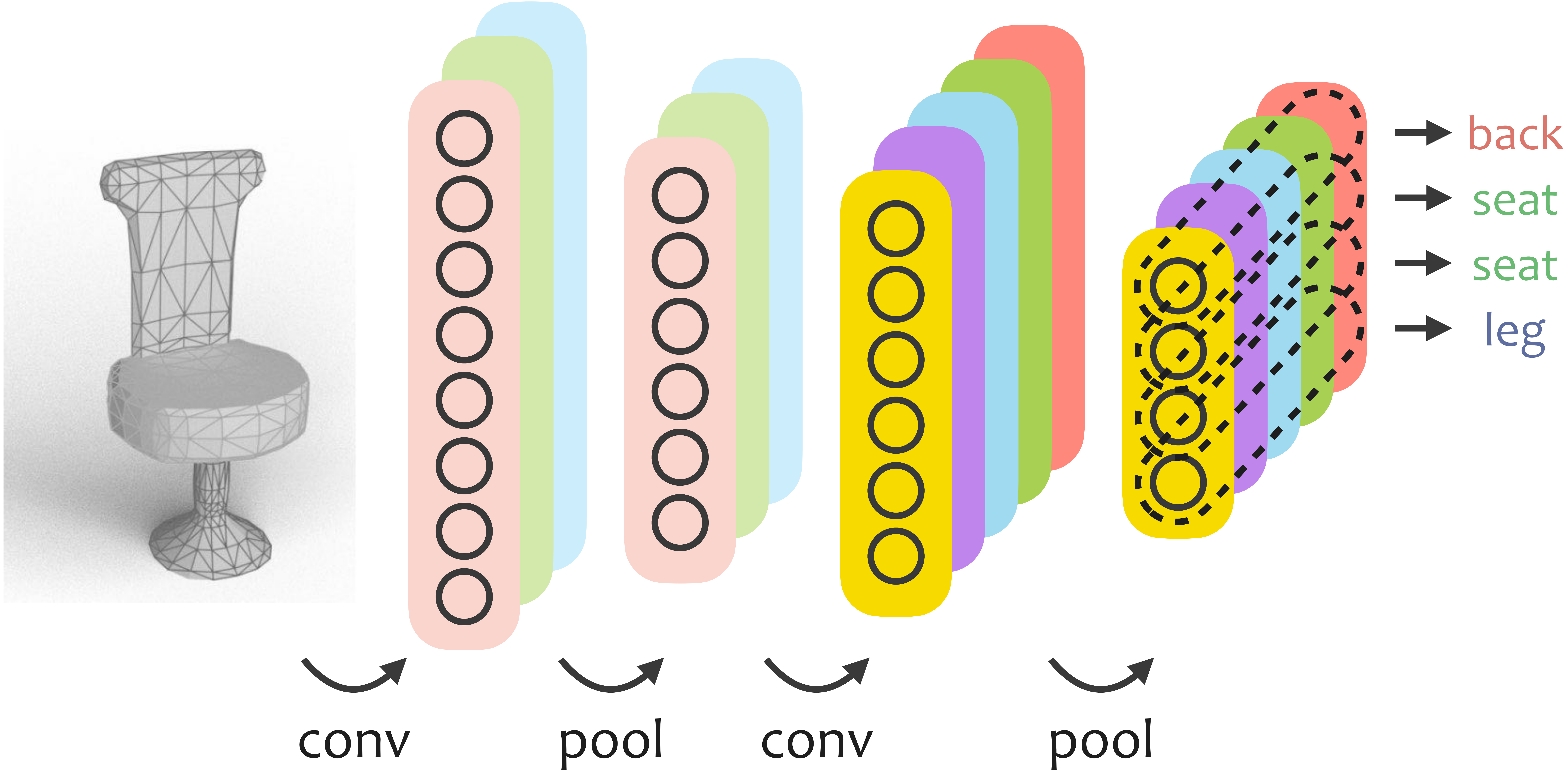
Learned Local Descriptors



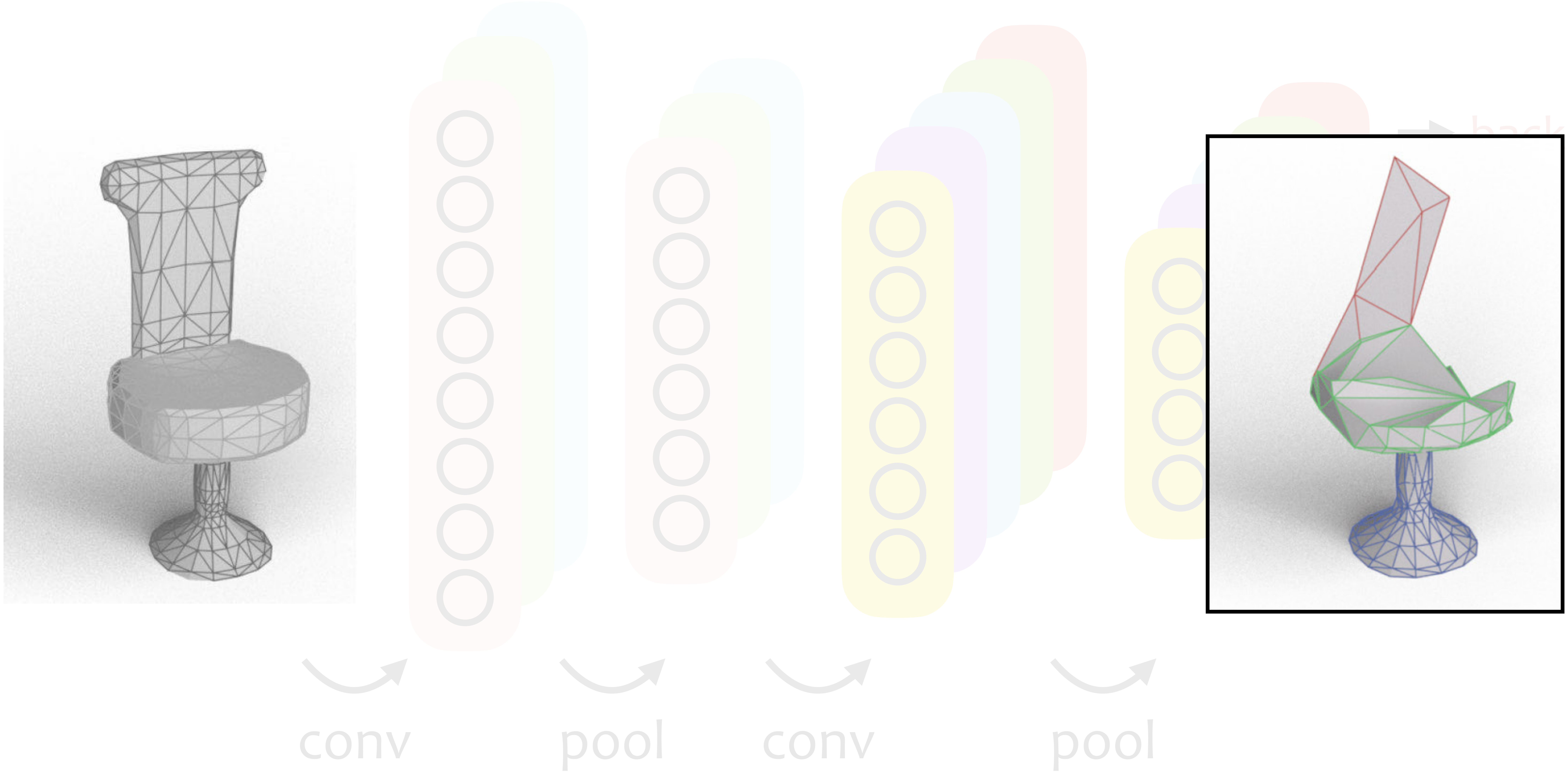
Learned Local Descriptors



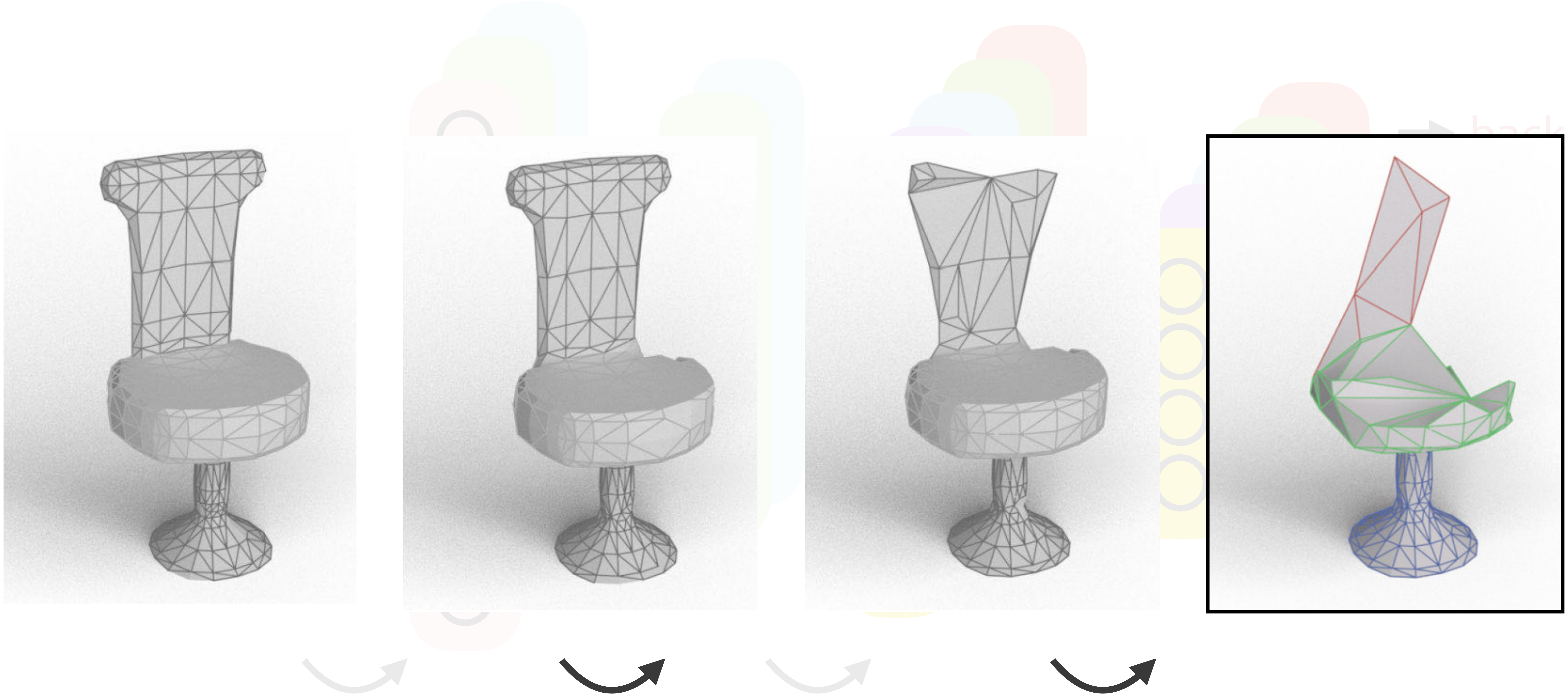
Learned Local Descriptors



Machine Learning Segmentation

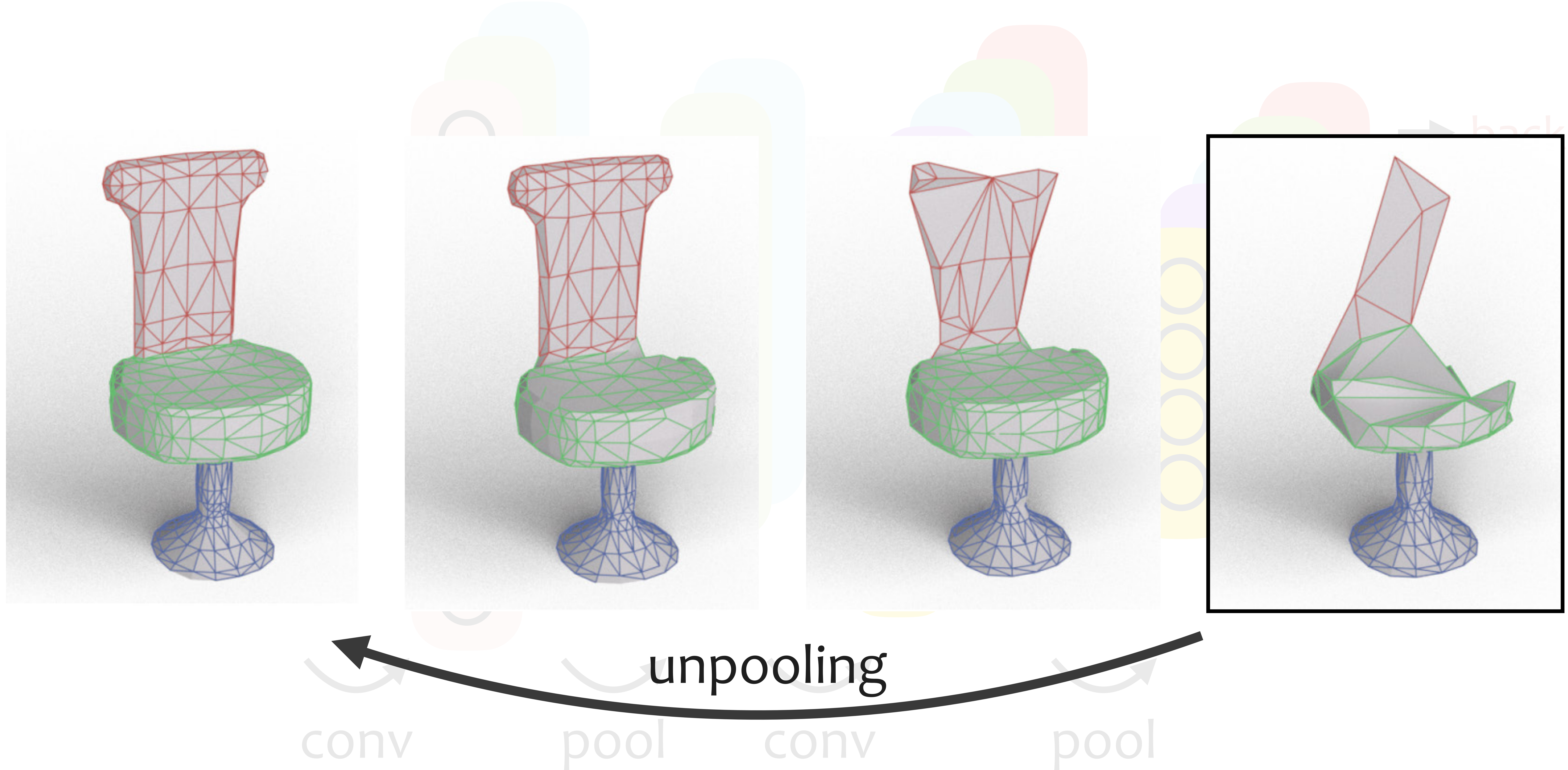


Machine Learning Segmentation

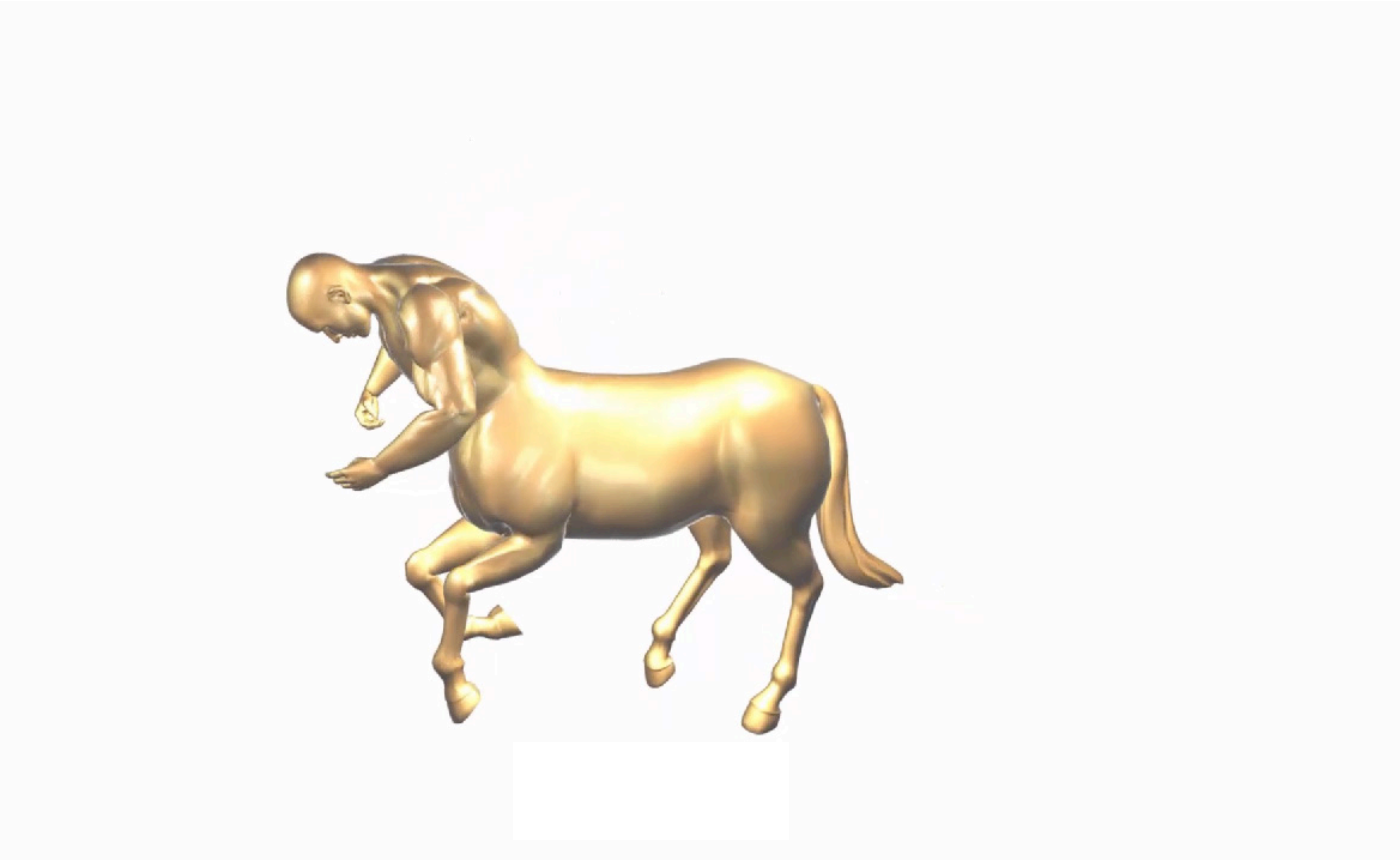


conv pool conv pool

Machine Learning Segmentation



Handle Shape Variants



isometry

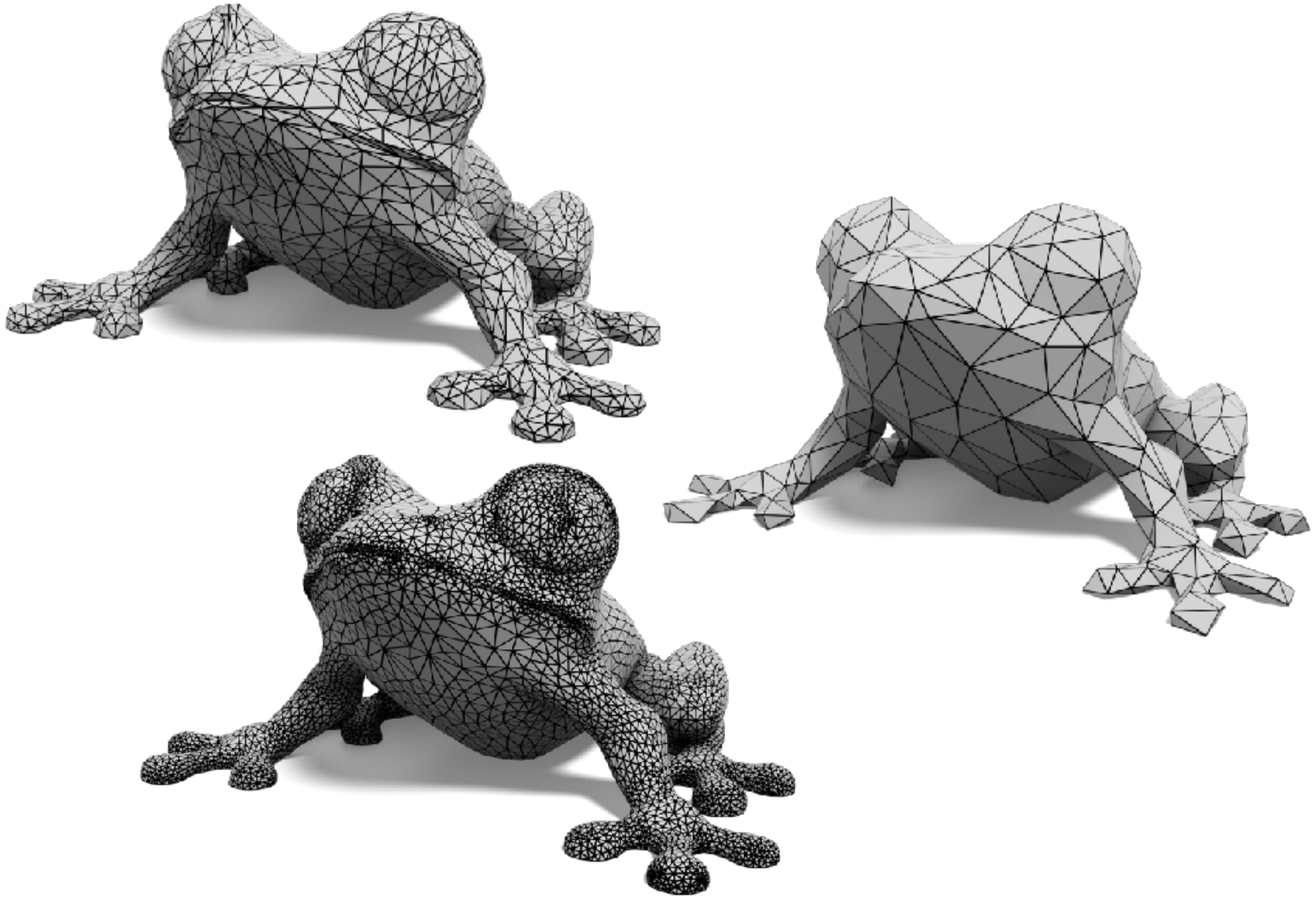
image source: Brandt et al. 2016

Handle Shape Variants

isometry invariant quantities

Handle Shape Variants

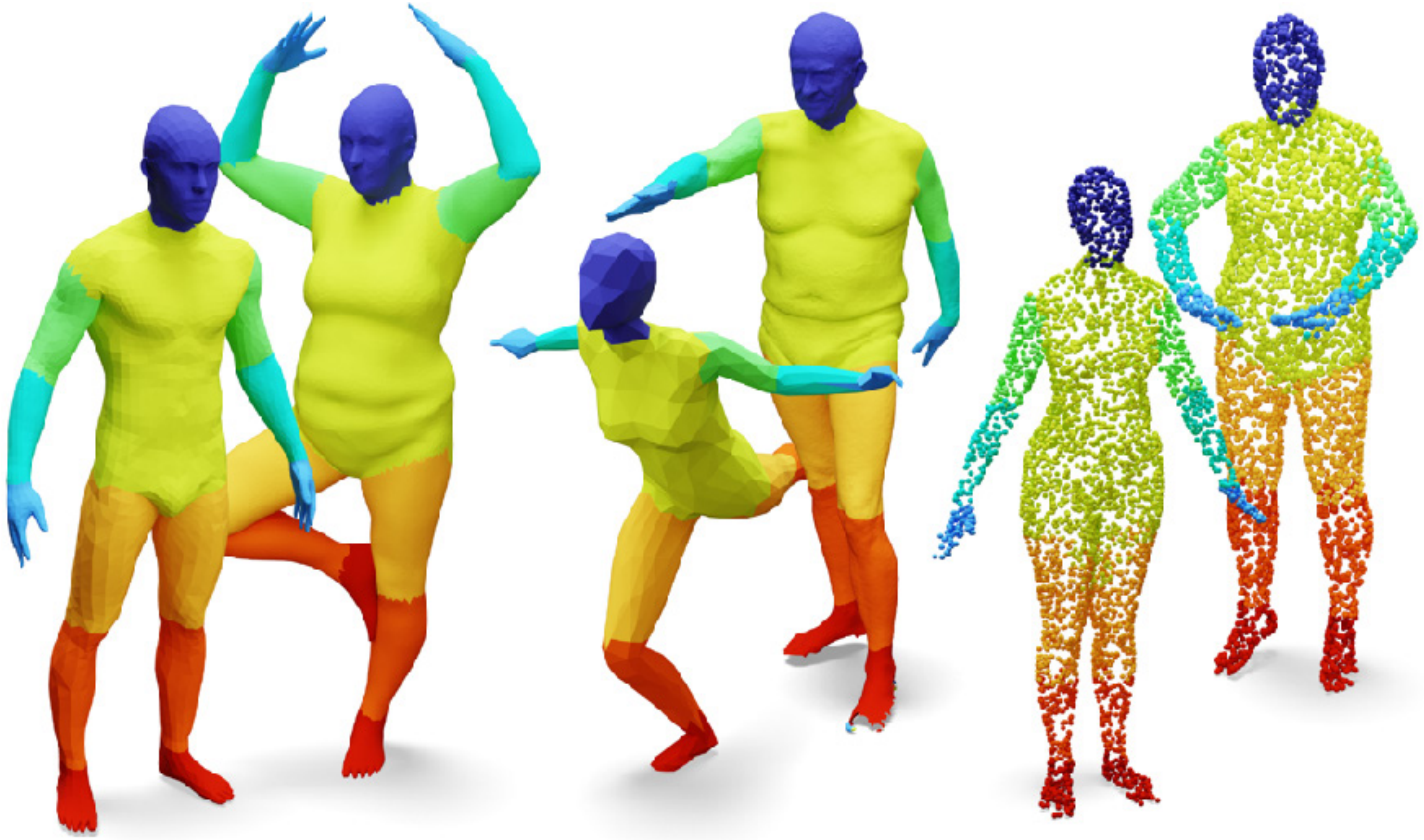
isometry invariant quantities



discretizations

Handle Shape Variants

isometry invariant quantities

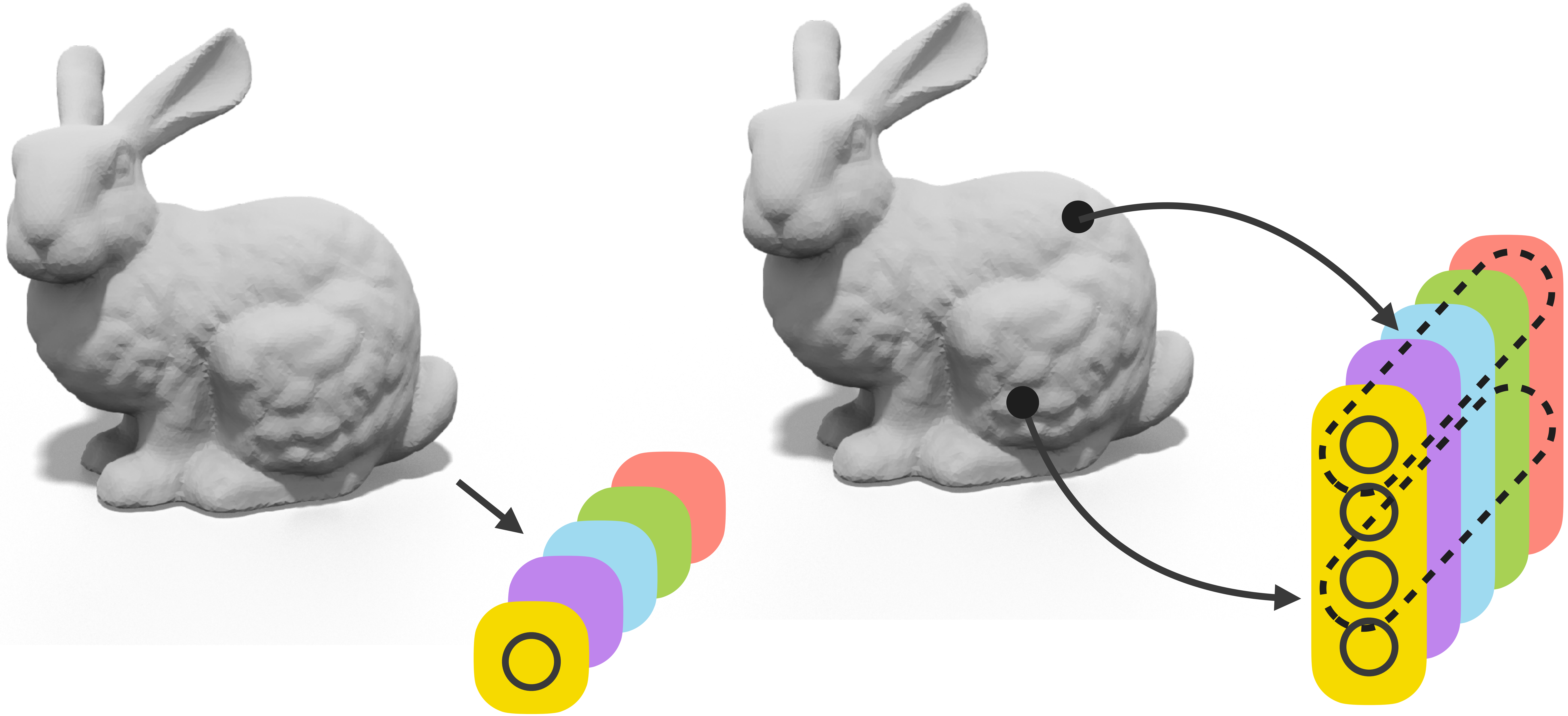


discretization agnostic convolution
(e.g., Sharp et al. 2021)

Segmentation

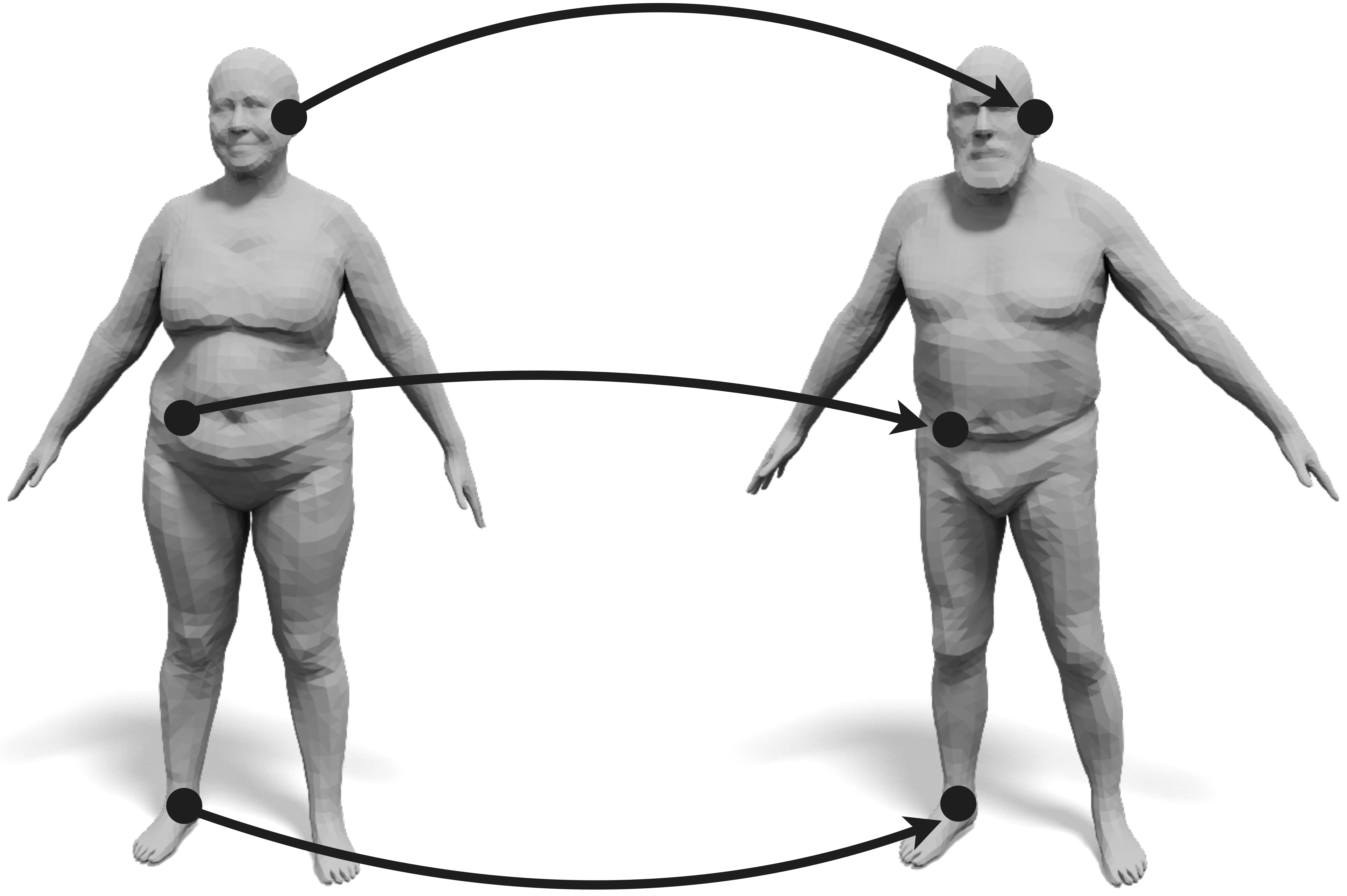
Smirnov & Solomon, 2021

ML as Feature Extractors

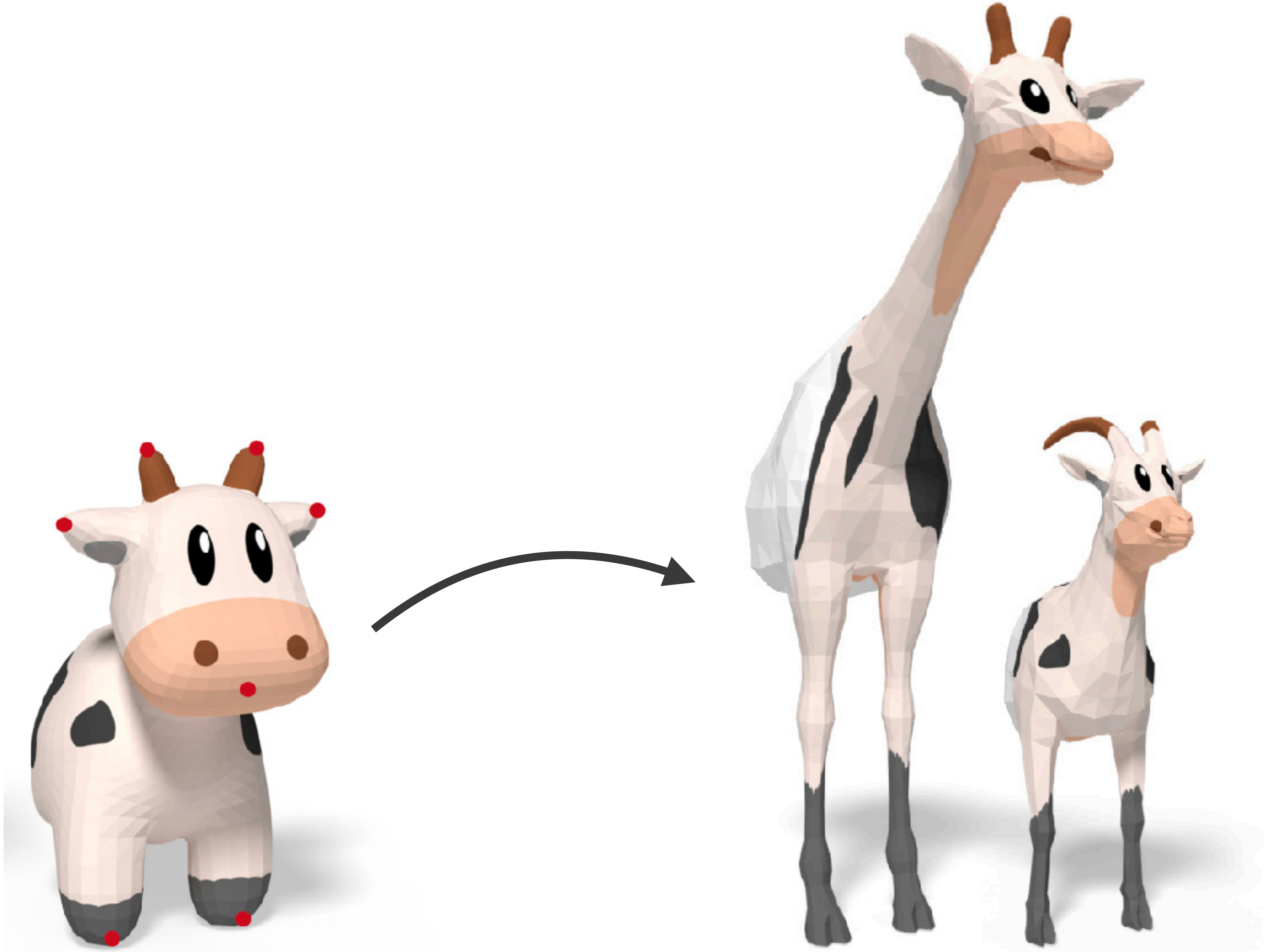


Applications of learned feature extractors
— combined with classic methods —

Example: Shape Matching

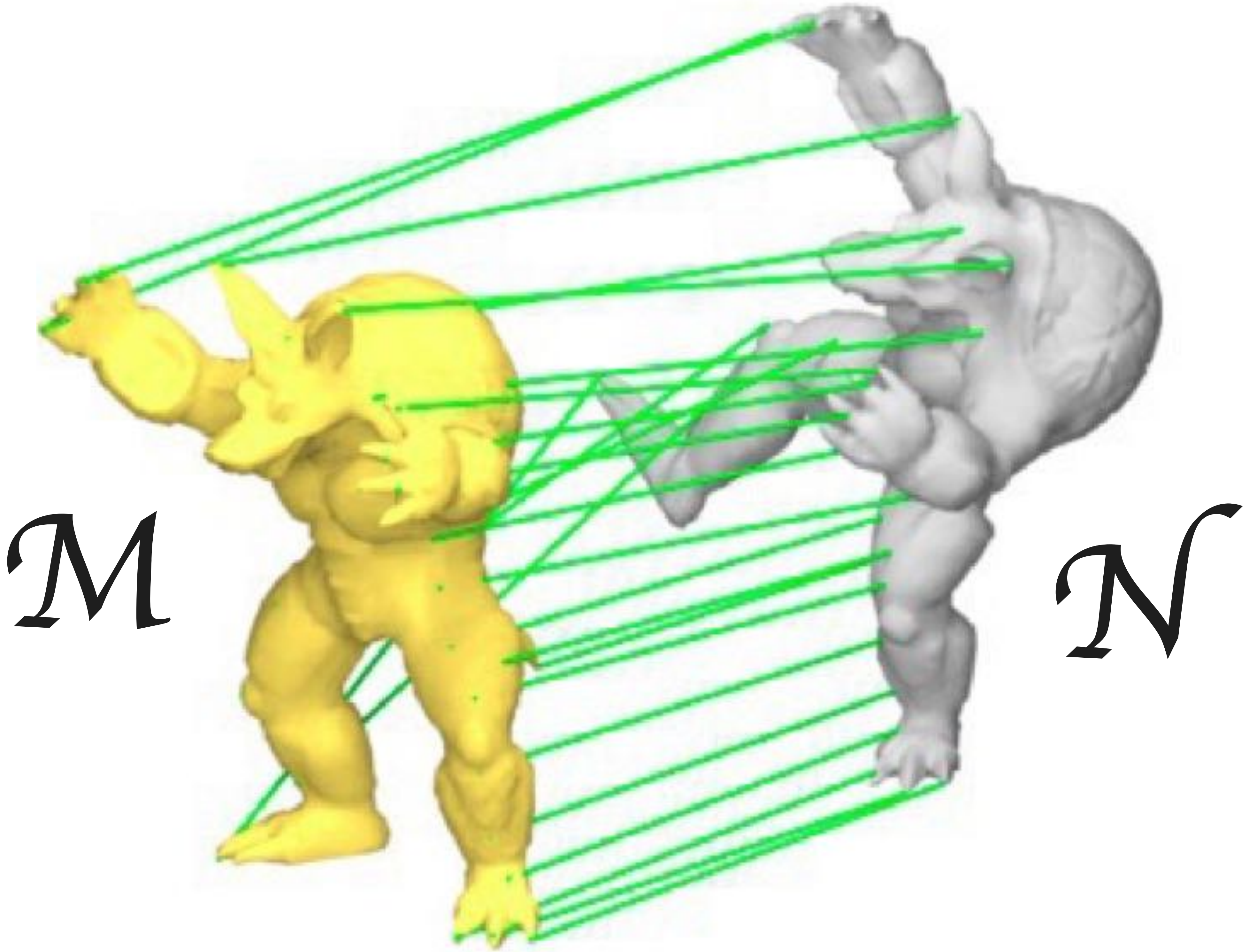


Texture Transfer



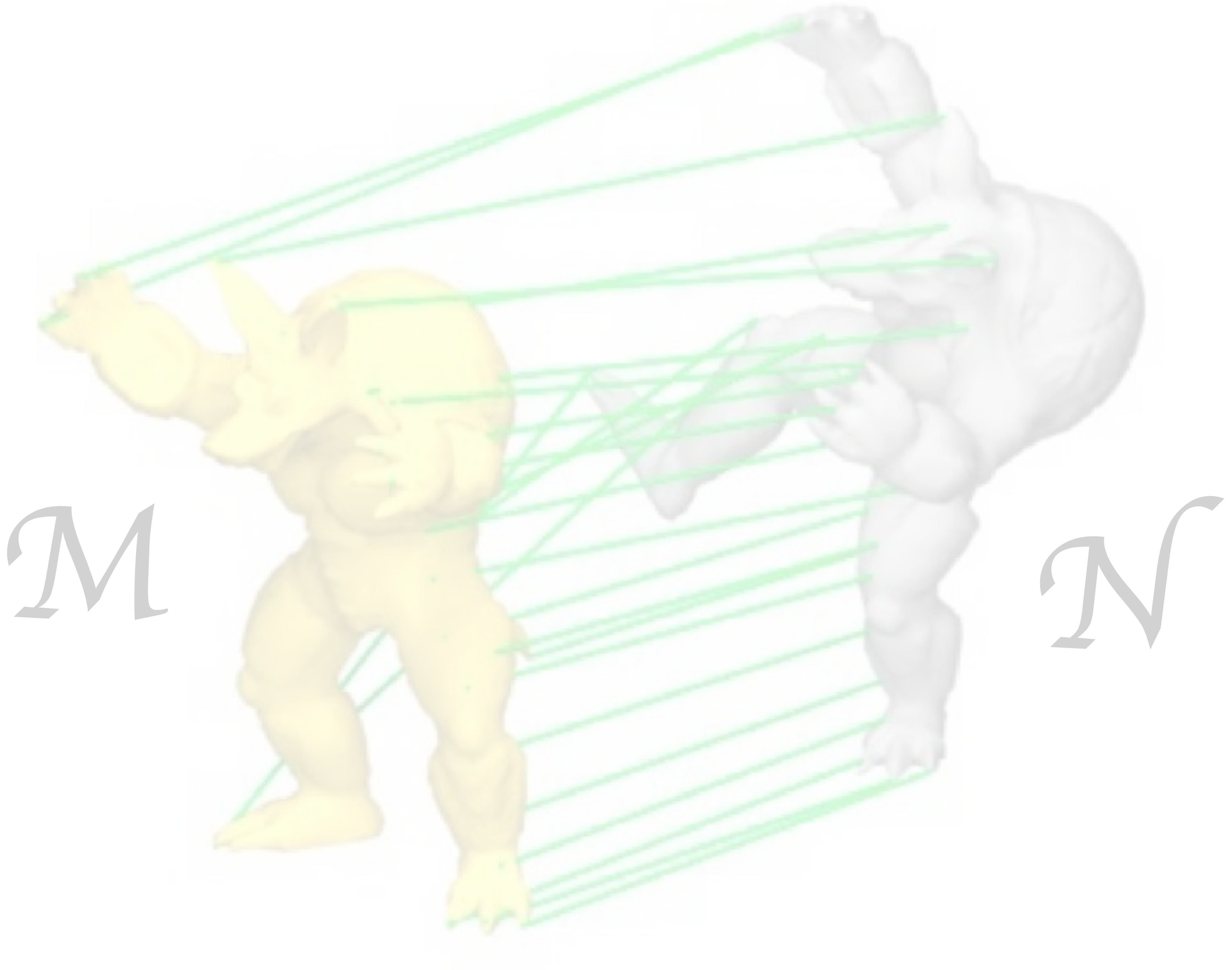
Schmidt et al. 2019

Point Maps

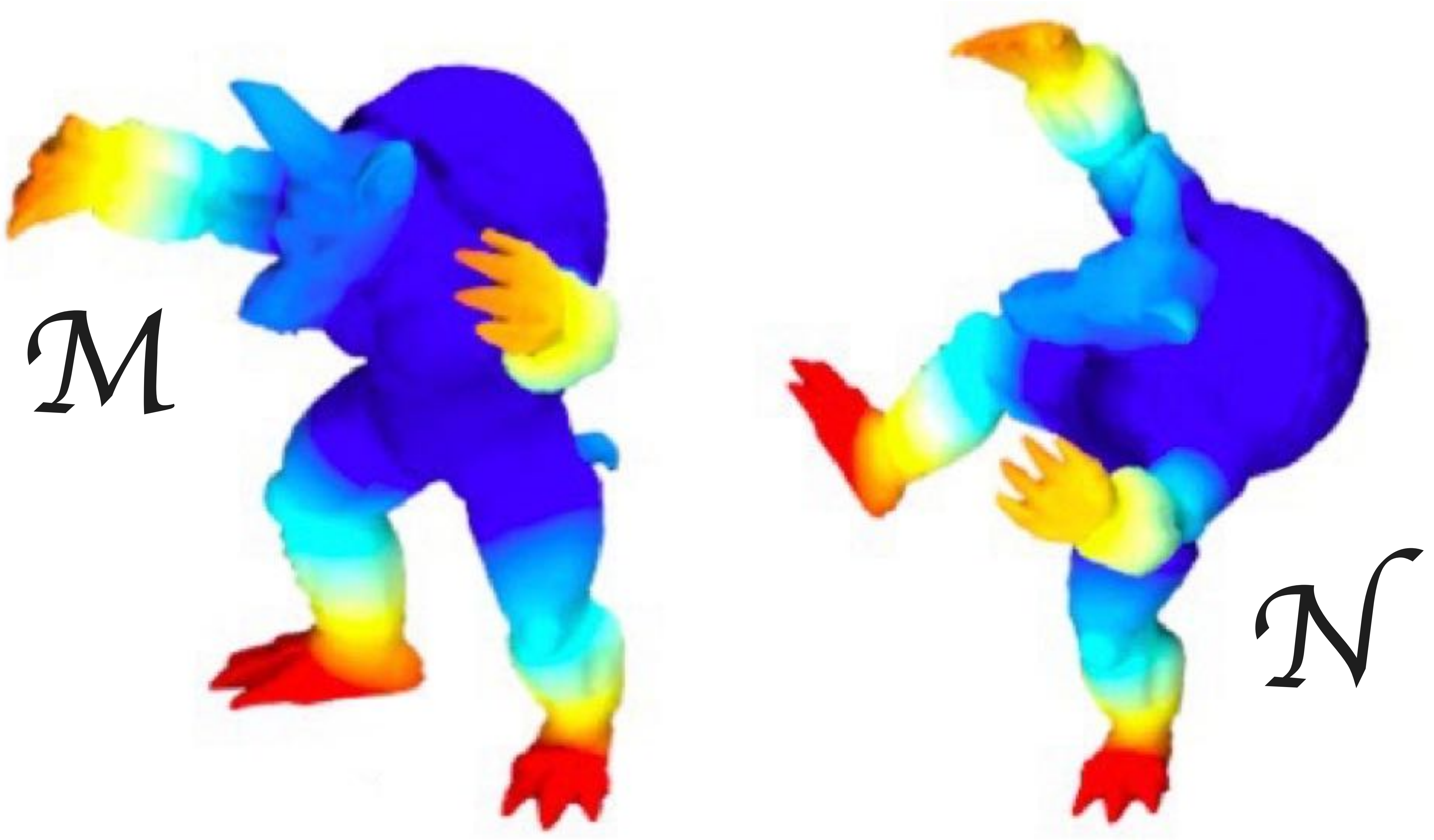


a point on $\mathcal{M}$ → a point on $\mathcal{N}$

Functional Maps



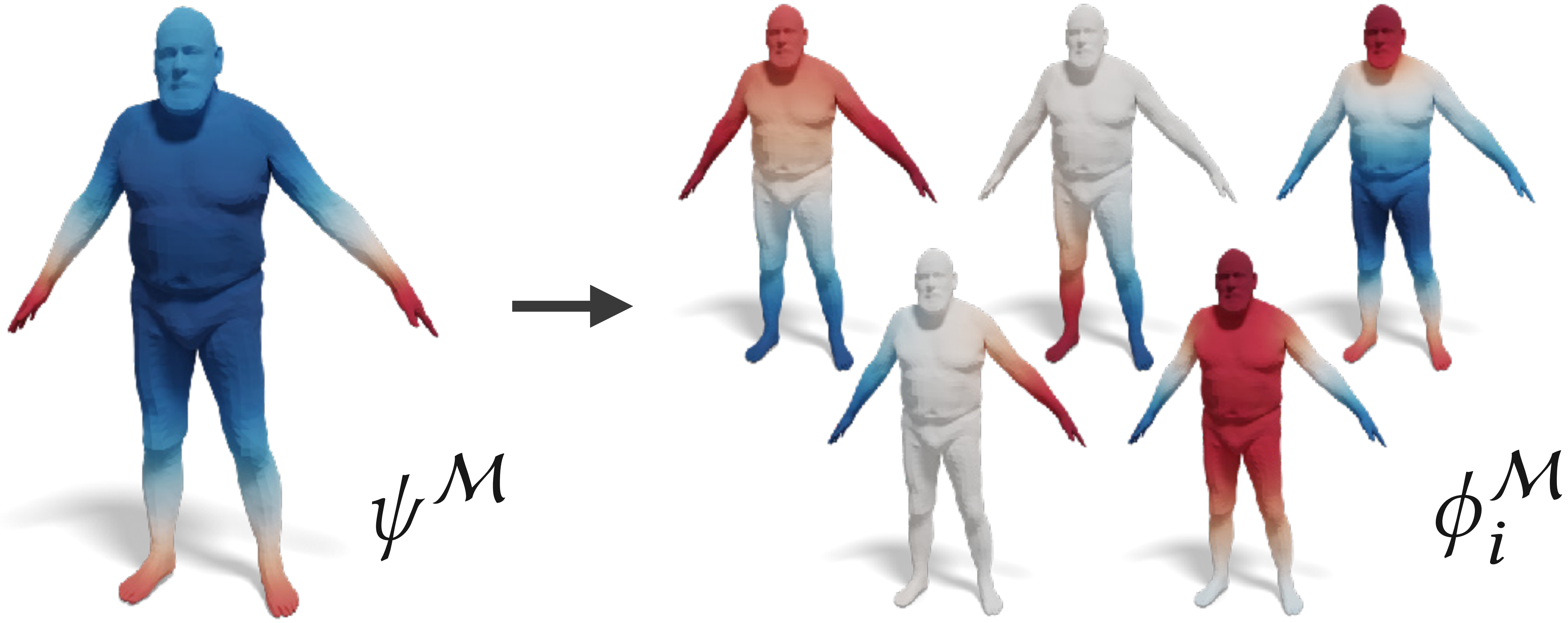
a point on $\mathcal{M}$ → a point on $\mathcal{N}$



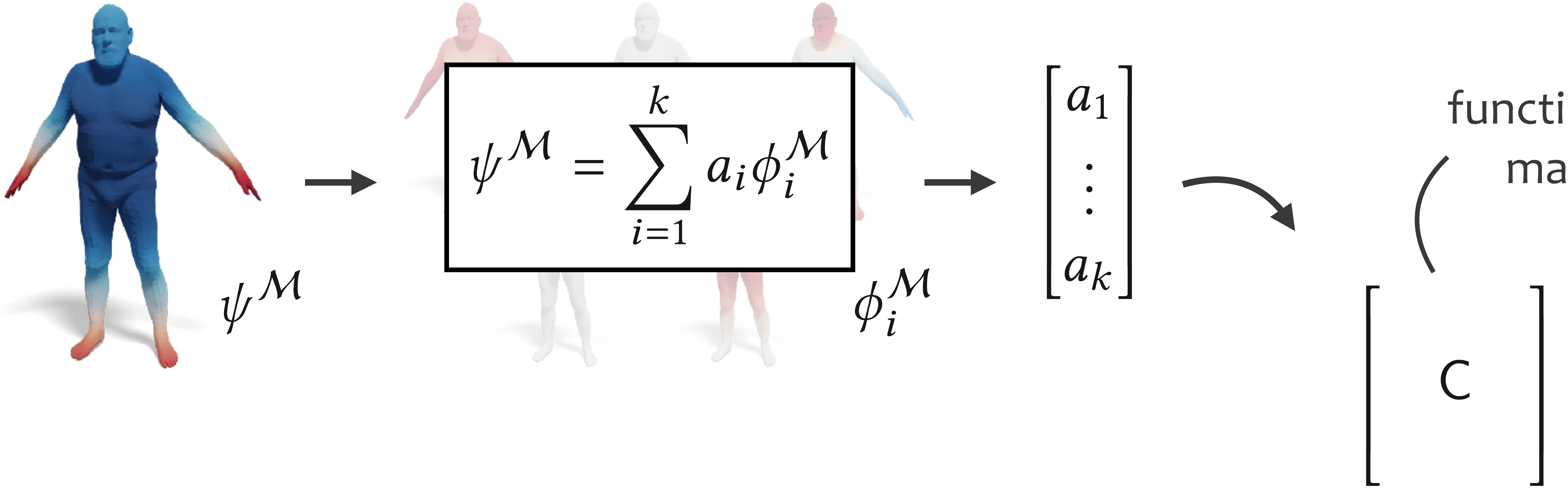
a function on $\mathcal{M}$ → a function on $\mathcal{N}$

Ovsjanikov et al. 2012

Functional Maps Overview

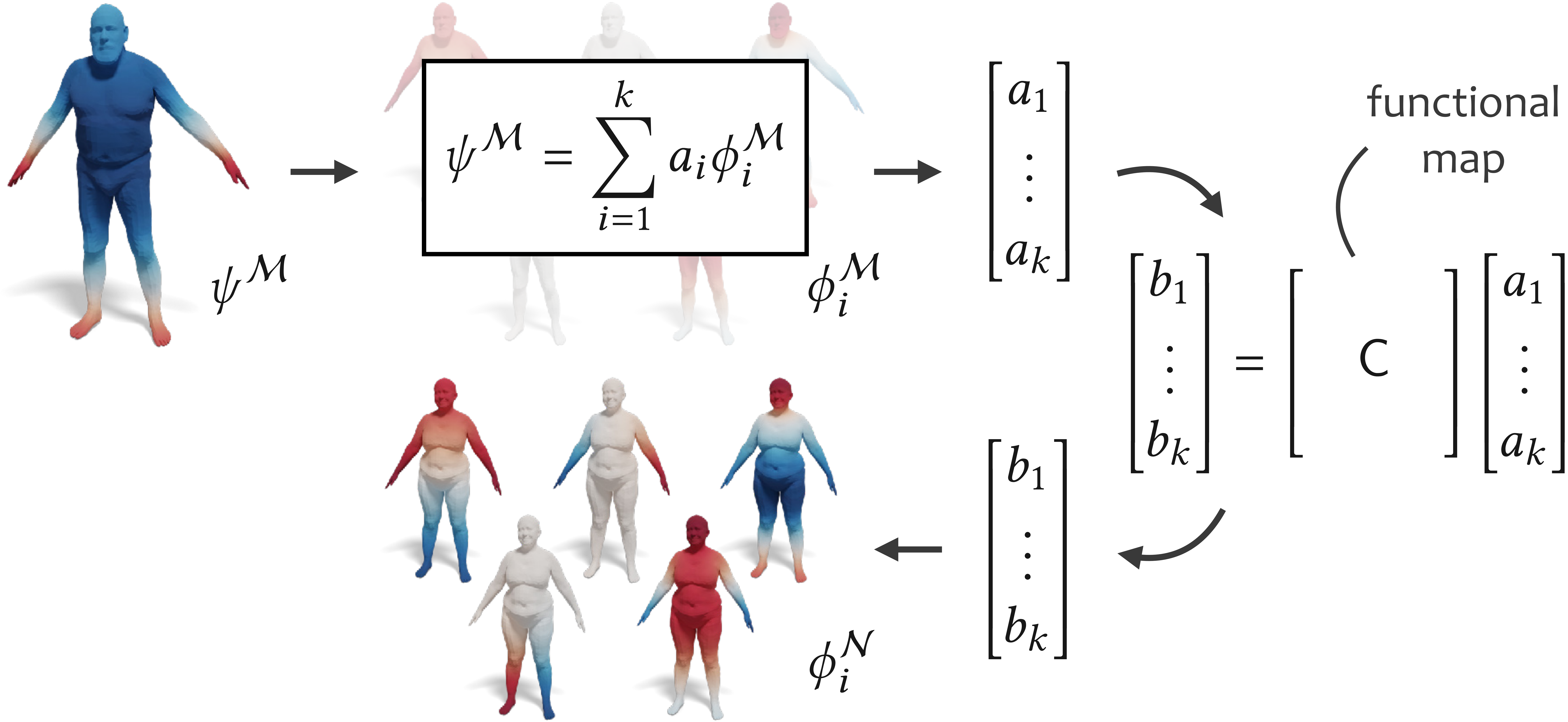


Functional Maps Overview



$$\psi^{\mathcal{M}} = \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$$

Functional Maps Overview



$\psi^{\mathcal{M}}$

$$\boxed{\psi^{\mathcal{M}} = \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}}$$

$\phi_i^{\mathcal{M}}$

$$\begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}$$

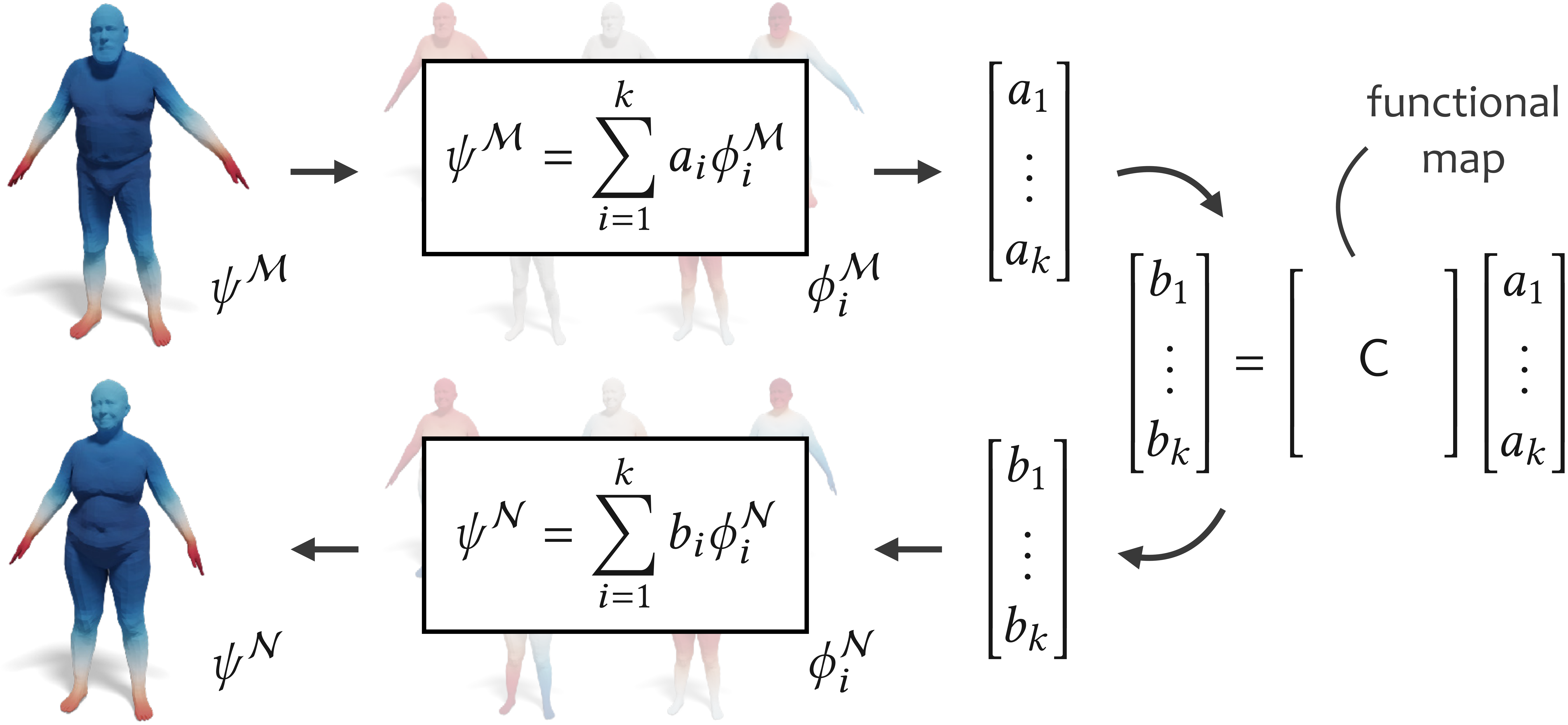
functional map

$$\begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} = \begin{bmatrix} & & \\ & \mathrm{C} & \\ & & \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}$$

$$\begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

$\phi_i^{\mathcal{N}}$

Functional Maps Overview



$$\psi^{\mathcal{M}} = \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$$

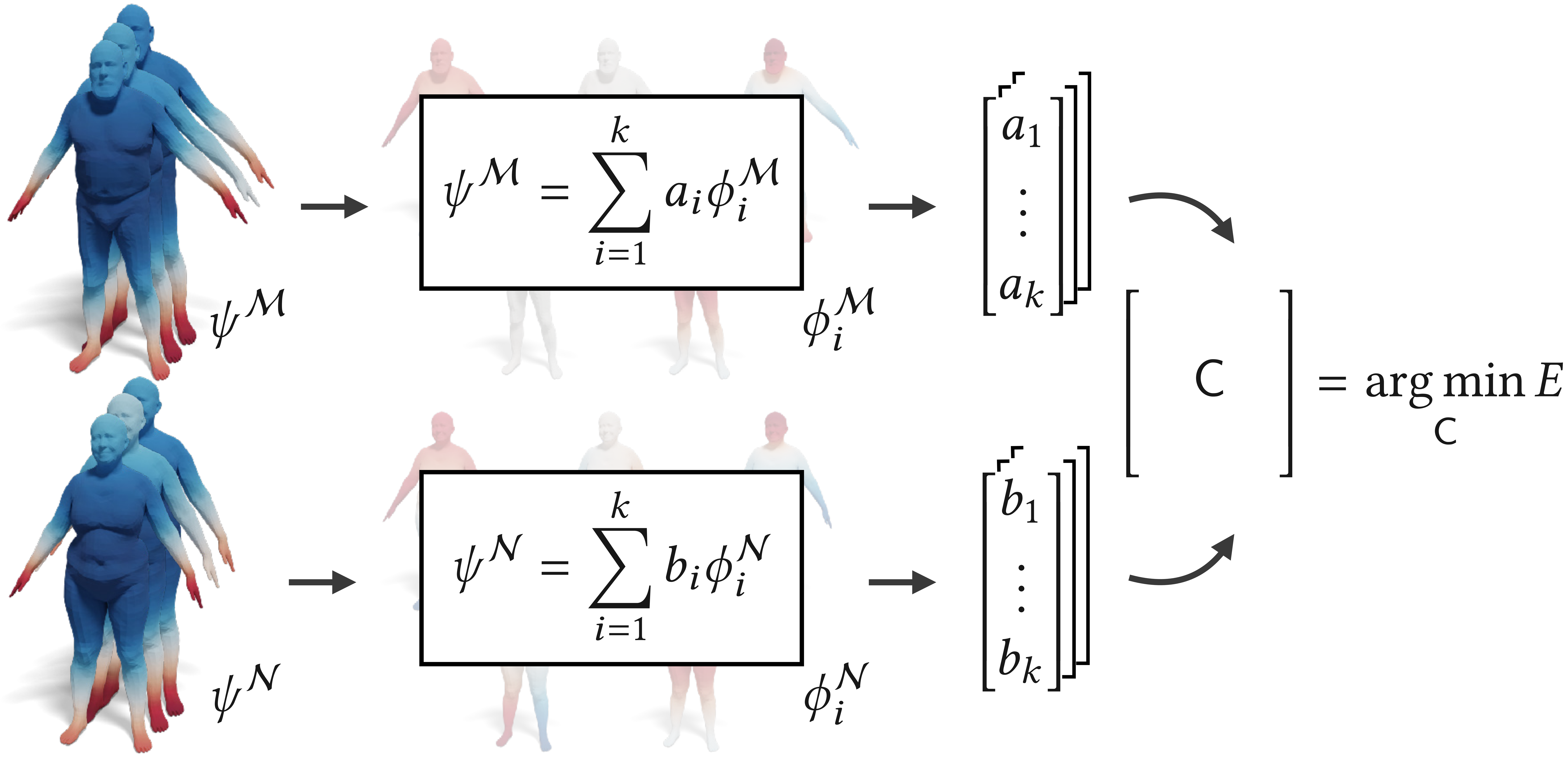
$$\begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}$$

$$\begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} = \begin{bmatrix} & C & \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}$$

$$\begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

$$\psi^{\mathcal{N}} = \sum_{i=1}^{k} b_i \phi_i^{\mathcal{N}}$$

Computing Functional Maps



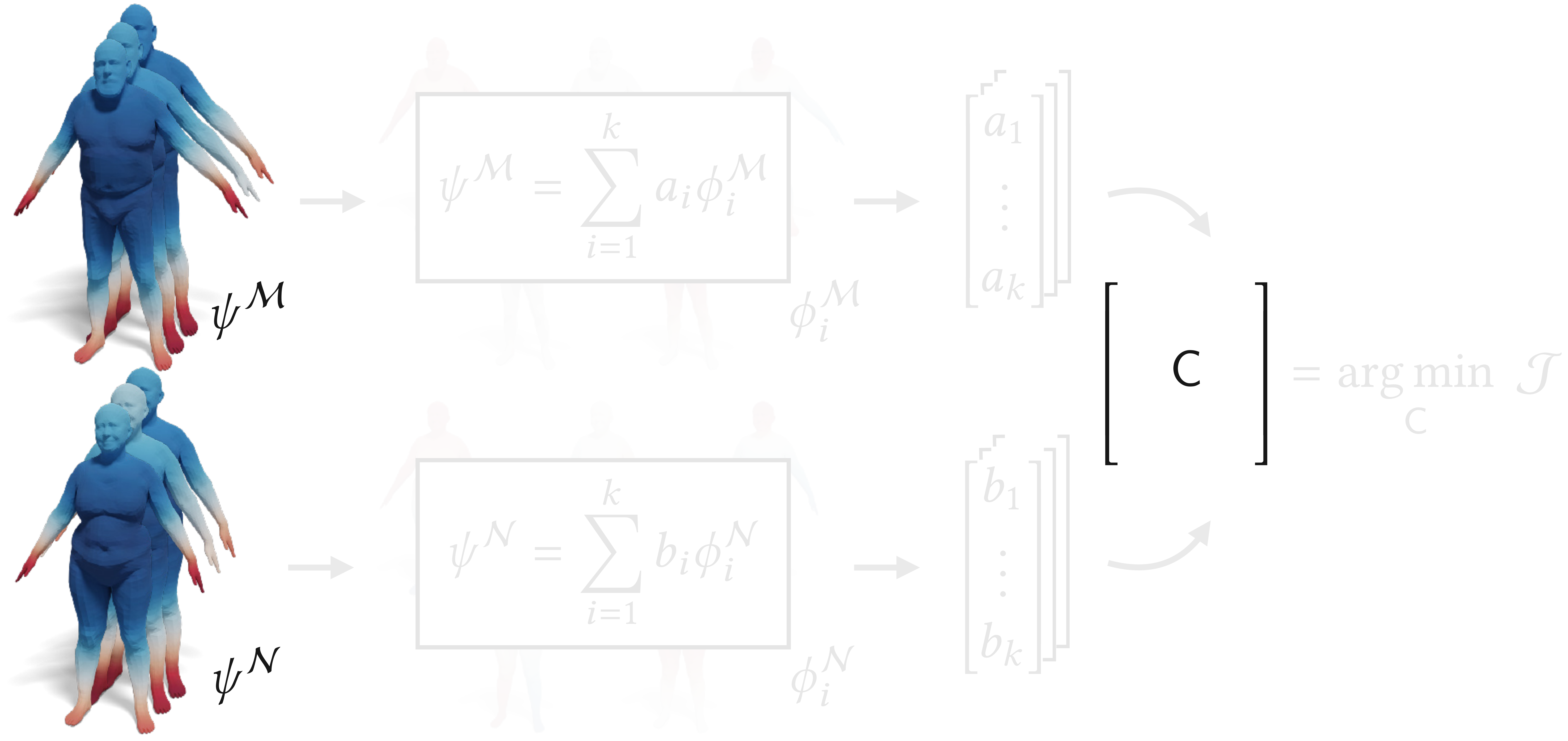
$$\psi^{\mathcal{M}} = \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$$

$$\psi^{\mathcal{N}} = \sum_{i=1}^{k} b_i \phi_i^{\mathcal{N}}$$

$$\begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} \qquad \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

$$\begin{bmatrix} \quad \mathrm{C} \quad \end{bmatrix} = \arg\min_{\mathrm{C}} E$$

Computing Functional Maps



$\psi^{\mathcal{M}}$

$$\psi^{\mathcal{M}} = \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$$

$\phi_i^{\mathcal{M}}$

$$\begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}$$

$$\begin{bmatrix} \quad C \quad \end{bmatrix} = \underset{C}{\arg\min}\ \mathcal{J}$$

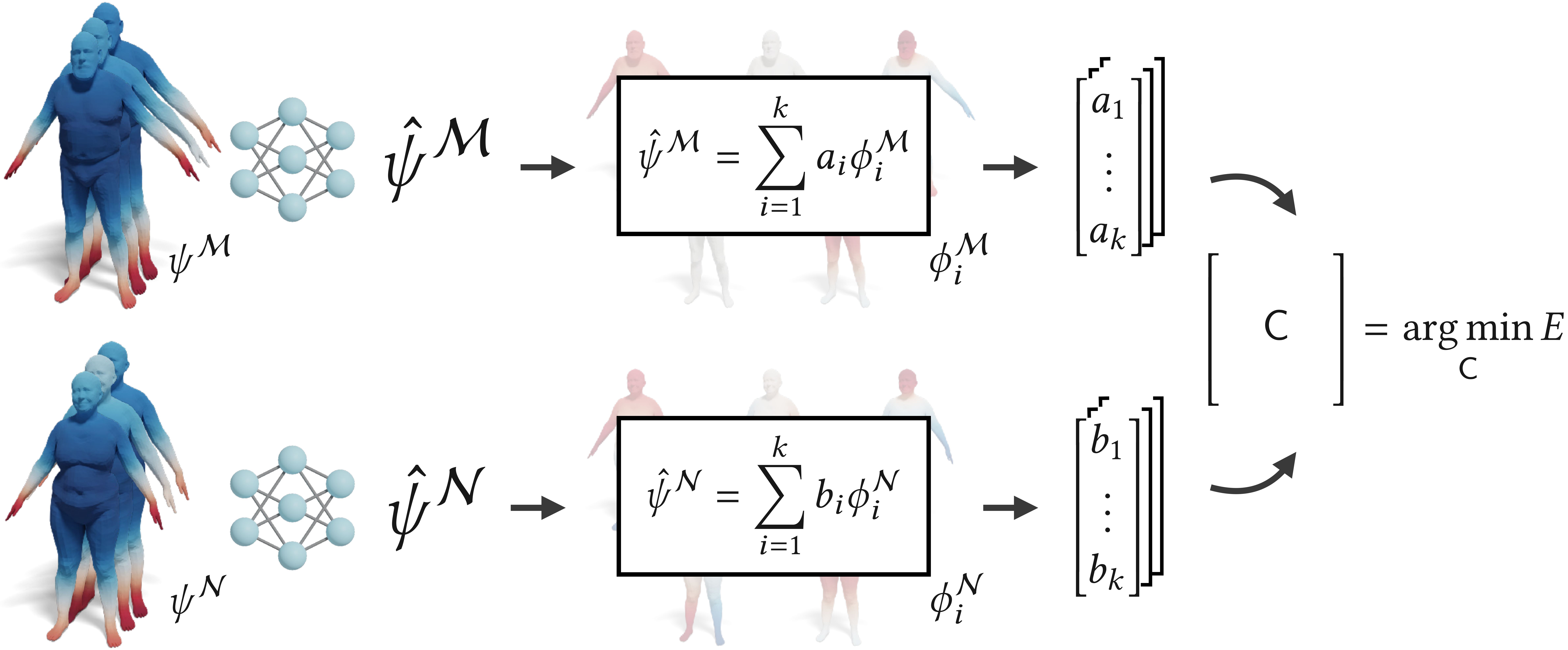
$\psi^{\mathcal{N}}$

$$\psi^{\mathcal{N}} = \sum_{i=1}^{k} b_i \phi_i^{\mathcal{N}}$$

$\phi_i^{\mathcal{N}}$

$$\begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

Deep Functional Maps



$$\hat{\psi}^{\mathcal{M}} = \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$$

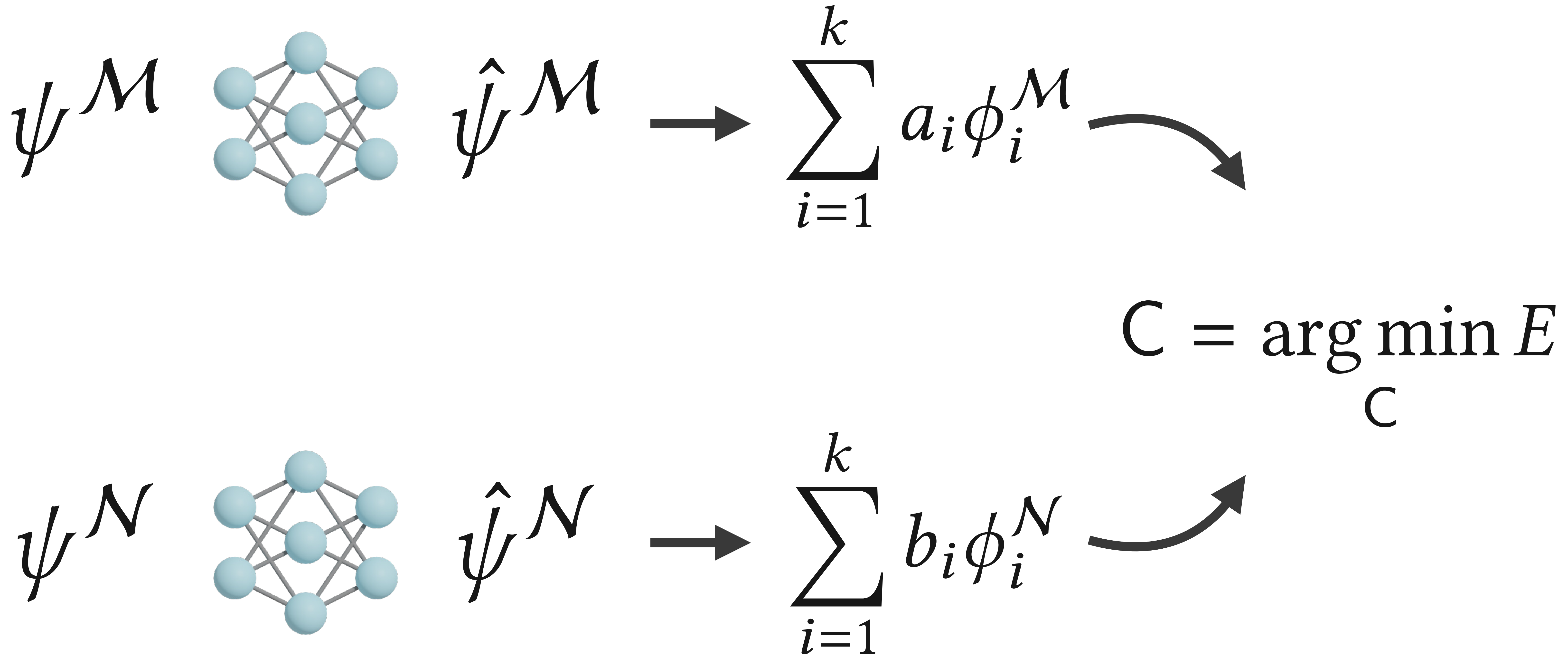
$$\hat{\psi}^{\mathcal{N}} = \sum_{i=1}^{k} b_i \phi_i^{\mathcal{N}}$$

$$\begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} \quad \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

$$\begin{bmatrix} \mathrm{C} \end{bmatrix} = \arg\min_{\mathrm{C}} E$$

Litany et al. 2017

Deep Functional Maps



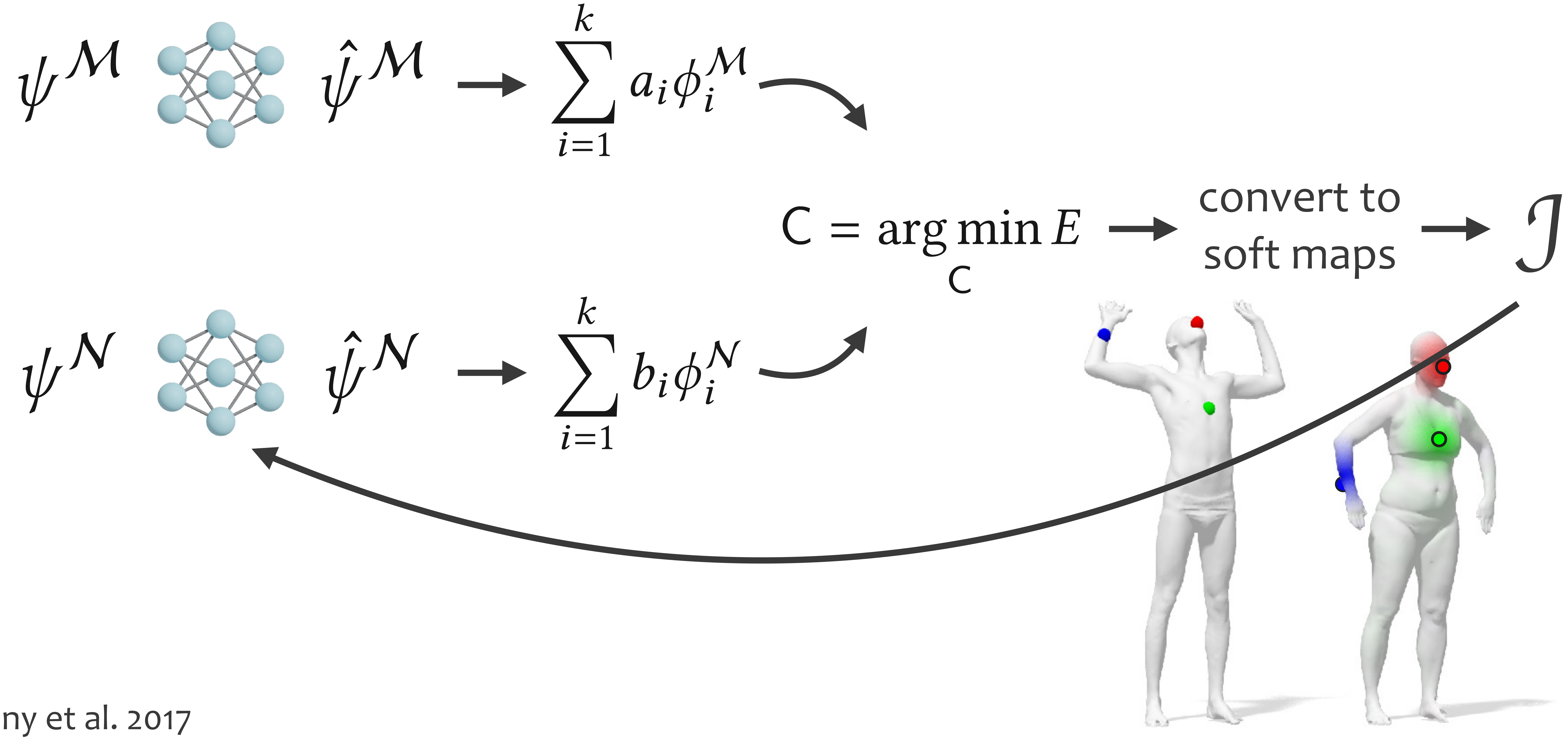
$$\psi^{\mathcal{M}} \quad \hat{\psi}^{\mathcal{M}} \rightarrow \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$$

$$\mathrm{C} = \arg\min_{\mathrm{C}} E$$

$$\psi^{\mathcal{N}} \quad \hat{\psi}^{\mathcal{N}} \rightarrow \sum_{i=1}^{k} b_i \phi_i^{\mathcal{N}}$$

Litany et al. 2017

Deep Functional Maps



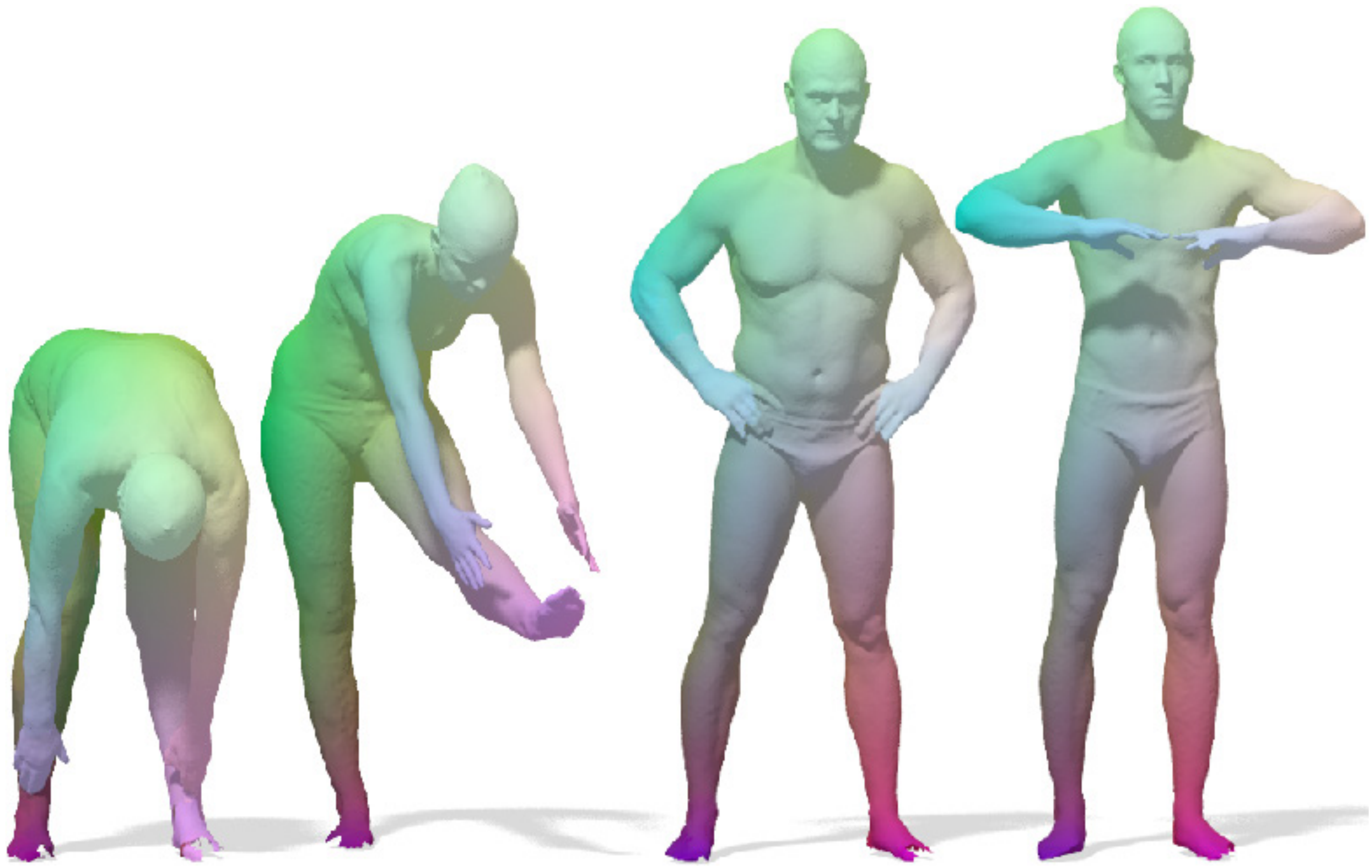
$\psi^{\mathcal{M}}$ $\quad \hat{\psi}^{\mathcal{M}} \rightarrow \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$

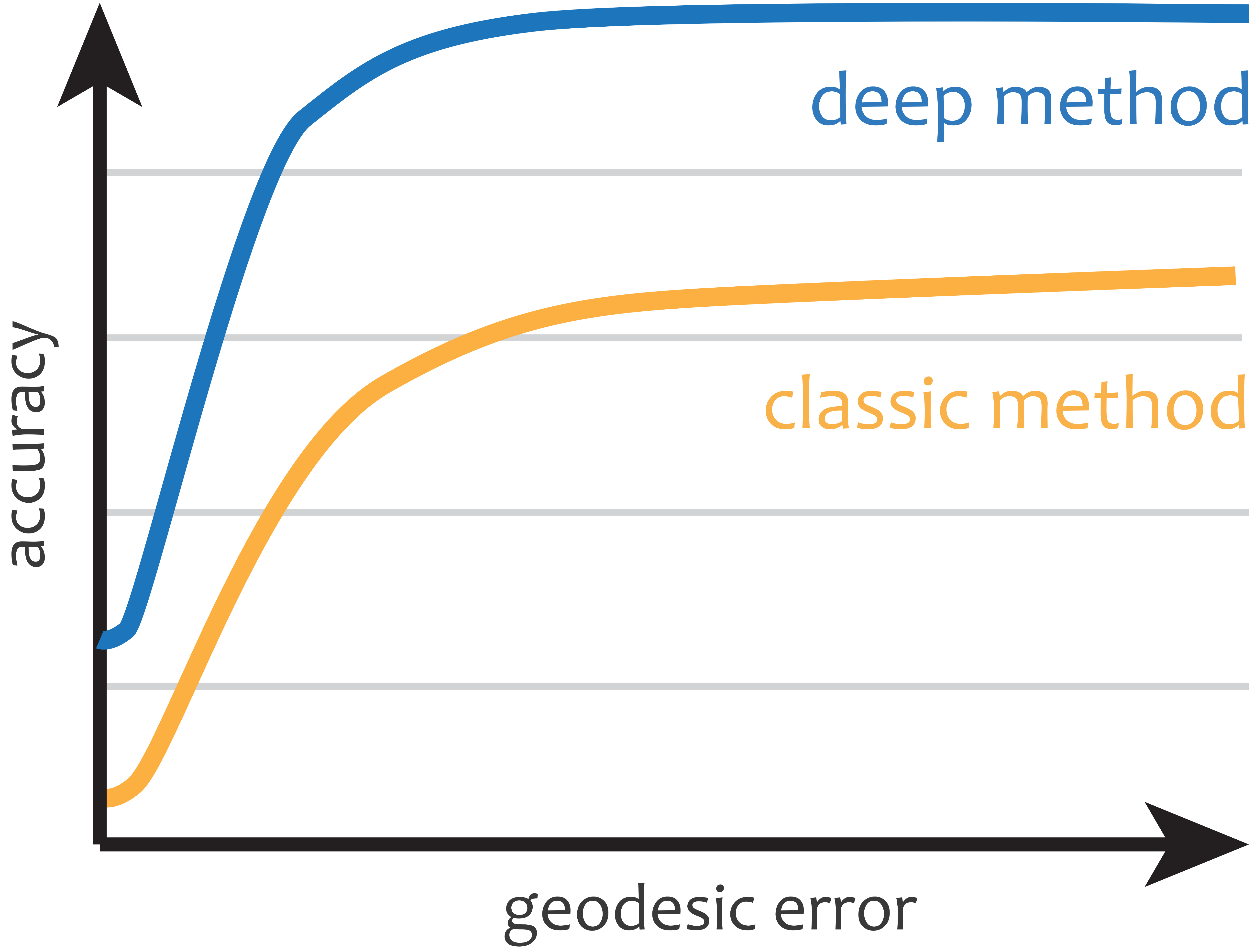
$\psi^{\mathcal{N}}$ $\quad \hat{\psi}^{\mathcal{N}} \rightarrow \sum_{i=1}^{k} b_i \phi_i^{\mathcal{N}}$

$\mathrm{C} = \arg\min_{\mathrm{C}} E$ → convert to soft maps → $\mathcal{J}$

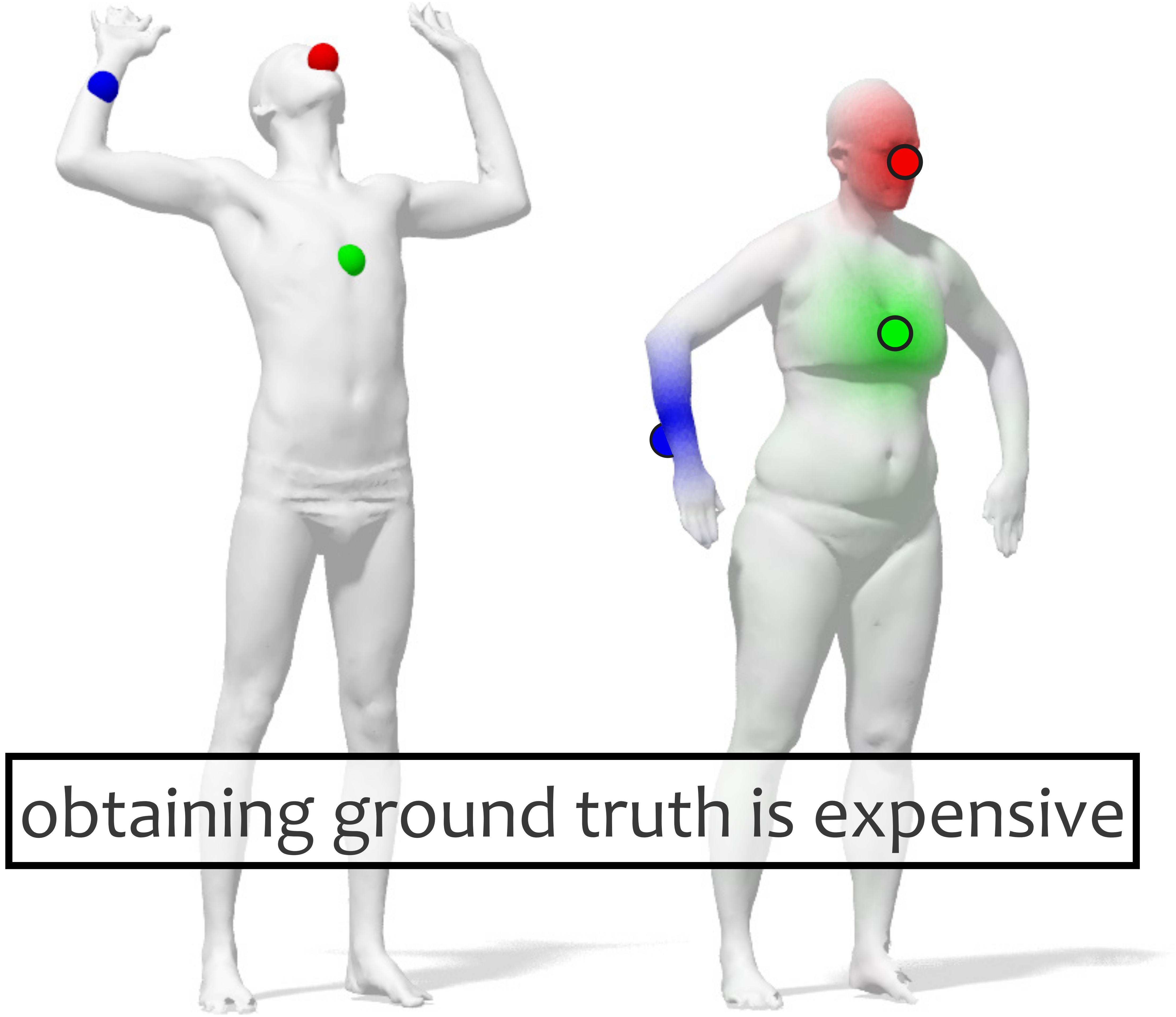
Litany et al. 2017

Push the limits





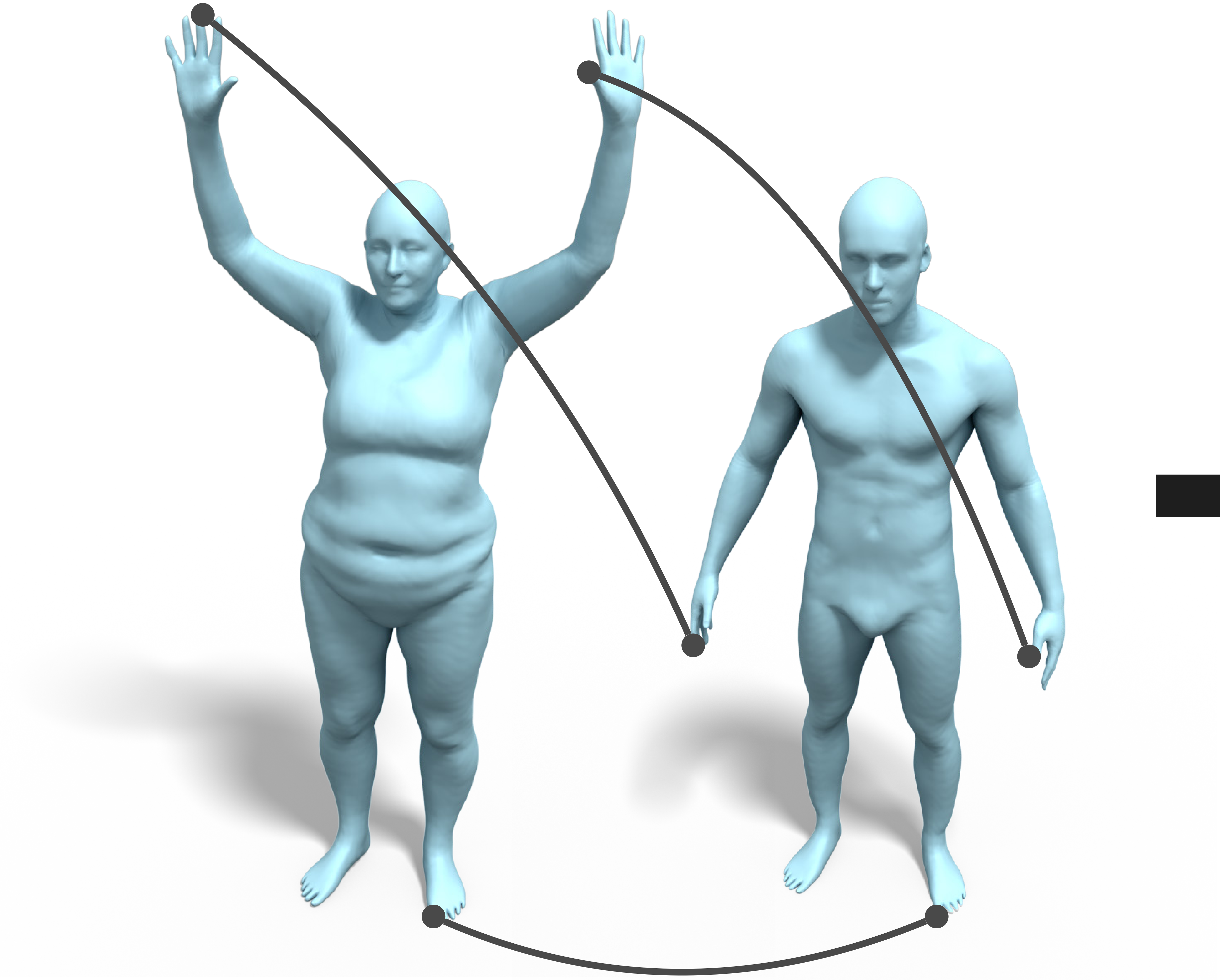
Requires Supervision



obtaining ground truth is expensive

Litany et al. 2017

Classic Approaches



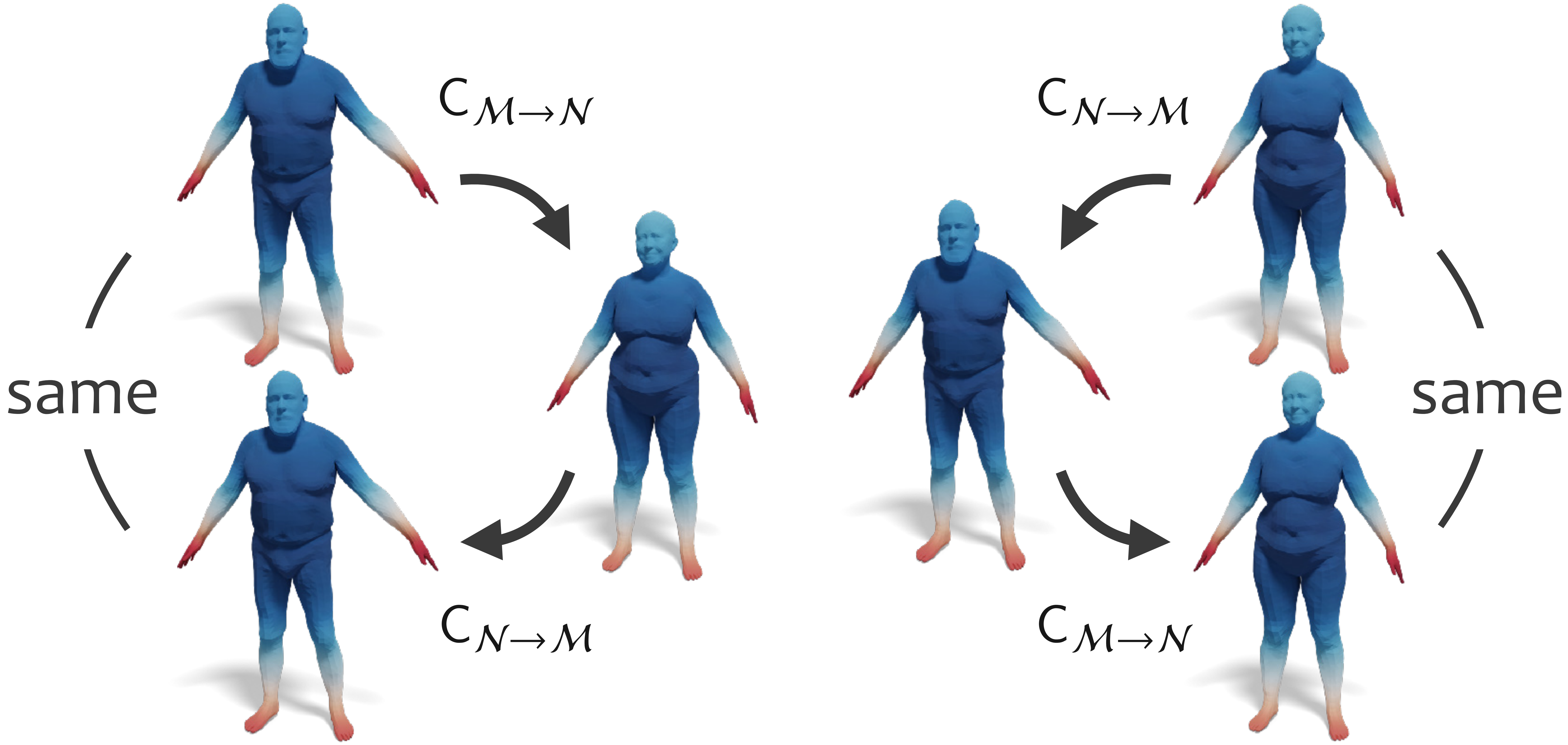
$$\underset{\mathrm{C}}{\arg\min}\, \mathcal{E}(\mathrm{C})$$

$$\begin{aligned}\mathcal{E}(\mathrm{C}) = \alpha_1 E_1(\mathrm{C}) + \alpha_2 E_2(\mathrm{C}) \\ + \alpha_3 E_3(\mathrm{C}) + \alpha_4 E_4(\mathrm{C})\end{aligned}$$

Ren et al. 2019

Desired Properties

- Bijectivity

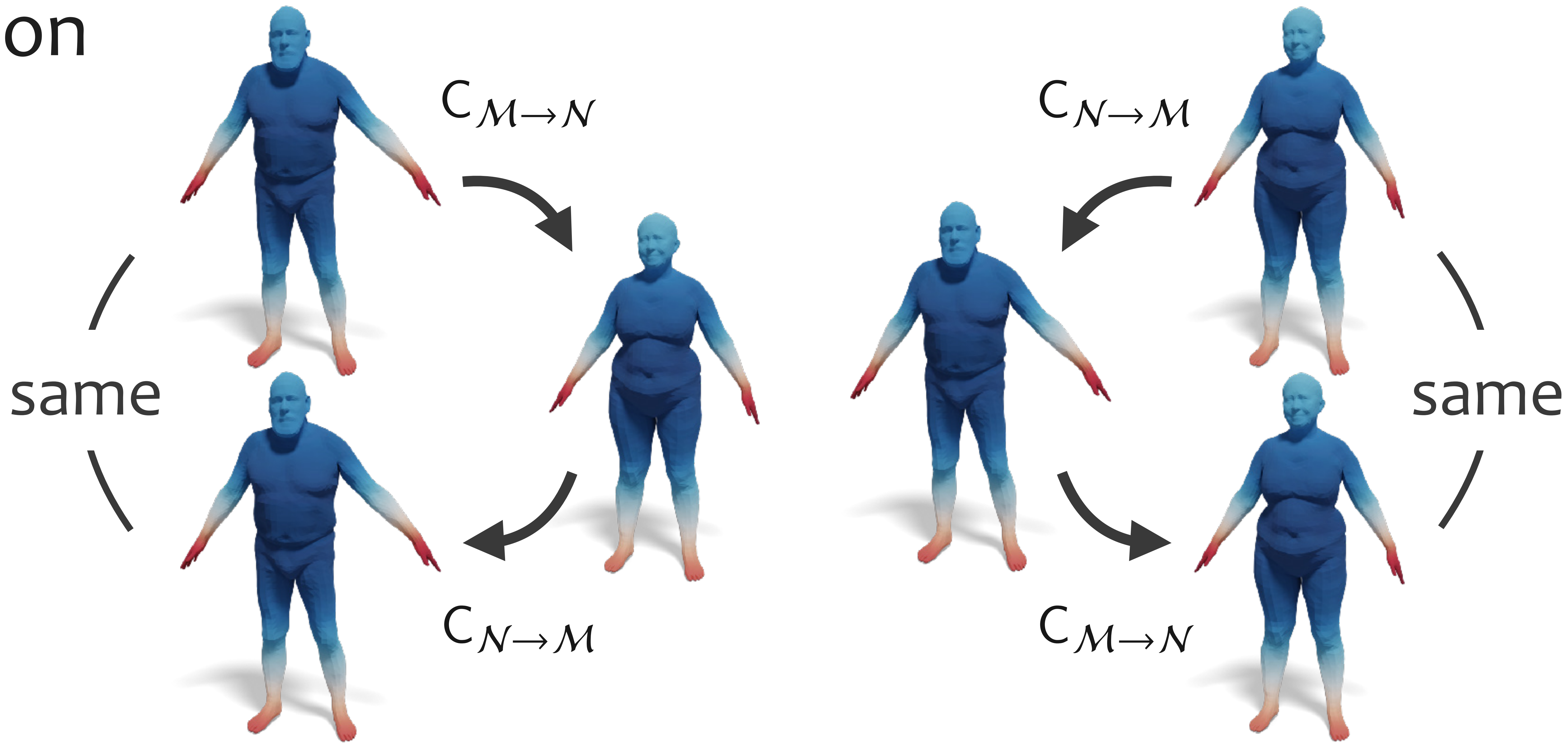


$C_{\mathcal{M}\to\mathcal{N}}$
$C_{\mathcal{N}\to\mathcal{M}}$
same
$C_{\mathcal{N}\to\mathcal{M}}$
$C_{\mathcal{M}\to\mathcal{N}}$
same

Ovsjanikov et al. 2016, 2017

Desired Properties

- Bijectivity
- Area preservation
- Laplacian commutativity
- Descriptor preservation
- …



$C_{\mathcal{M}\to\mathcal{N}}$
$C_{\mathcal{N}\to\mathcal{M}}$
same
$C_{\mathcal{N}\to\mathcal{M}}$
$C_{\mathcal{M}\to\mathcal{N}}$
same

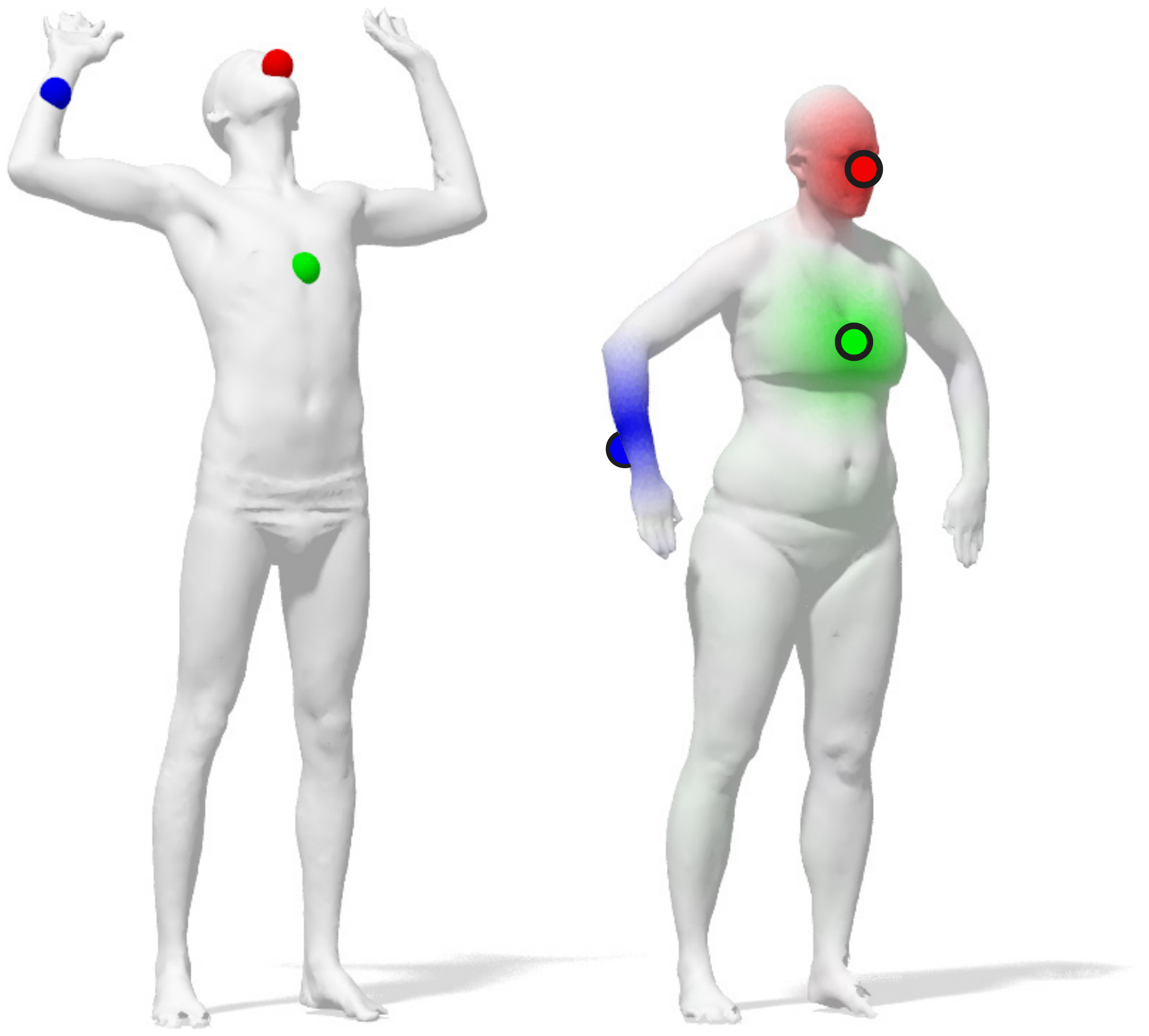
Ovsjanikov et al. 2016, 2017

From Supervised to Unsupervised

$$\psi^{\mathcal{M}} \quad \hat{\psi}^{\mathcal{M}} \rightarrow \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$$

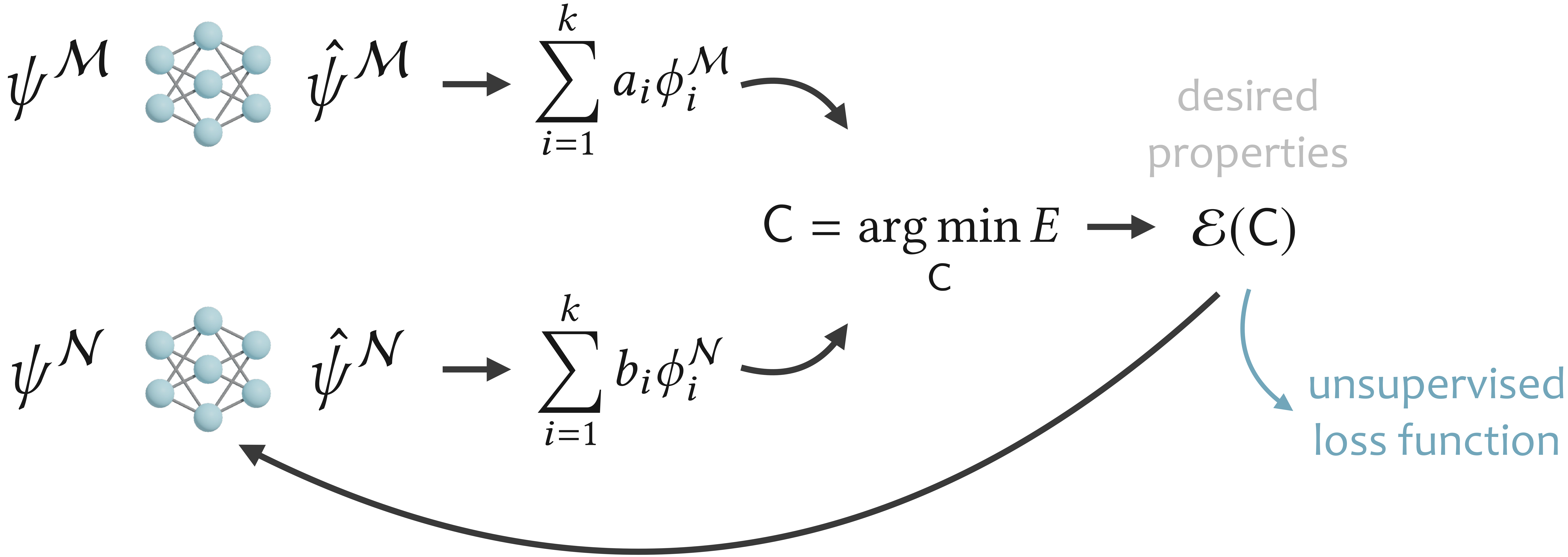
$$\psi^{\mathcal{N}} \quad \hat{\psi}^{\mathcal{N}} \rightarrow \sum_{i=1}^{k} b_i \phi_i^{\mathcal{N}}$$

$$\mathrm{C} = \arg\min_{\mathrm{C}} E \rightarrow \text{convert to soft maps} \rightarrow \mathcal{J}$$



Roufosse et al. 2019

From Supervised to Unsupervised



$\psi^{\mathcal{M}}$ $\hat{\psi}^{\mathcal{M}} \rightarrow \sum_{i=1}^{k} a_i \phi_i^{\mathcal{M}}$

$\psi^{\mathcal{N}}$ $\hat{\psi}^{\mathcal{N}} \rightarrow \sum_{i=1}^{k} b_i \phi_i^{\mathcal{N}}$

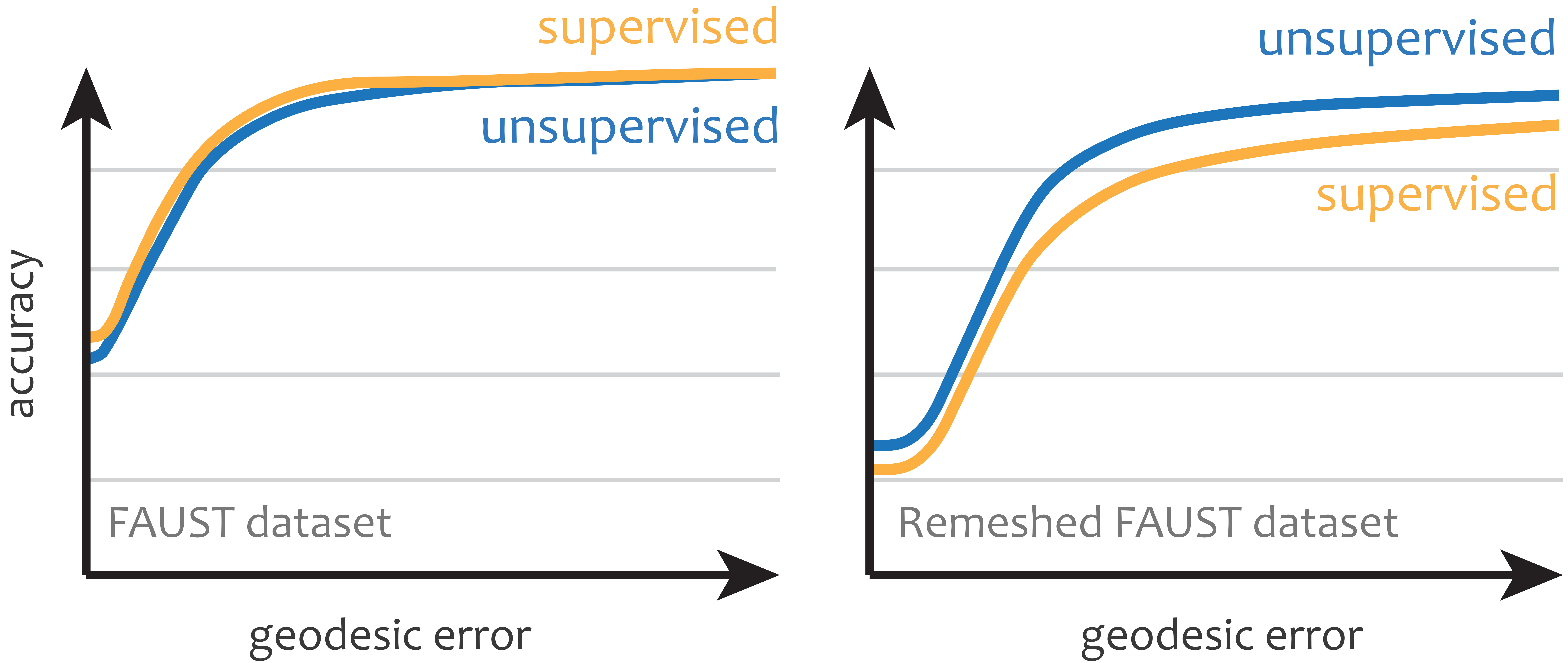
$\mathrm{C} = \arg\min_{\mathrm{C}} E \rightarrow \mathcal{E}(\mathrm{C})$

desired properties

unsupervised loss function

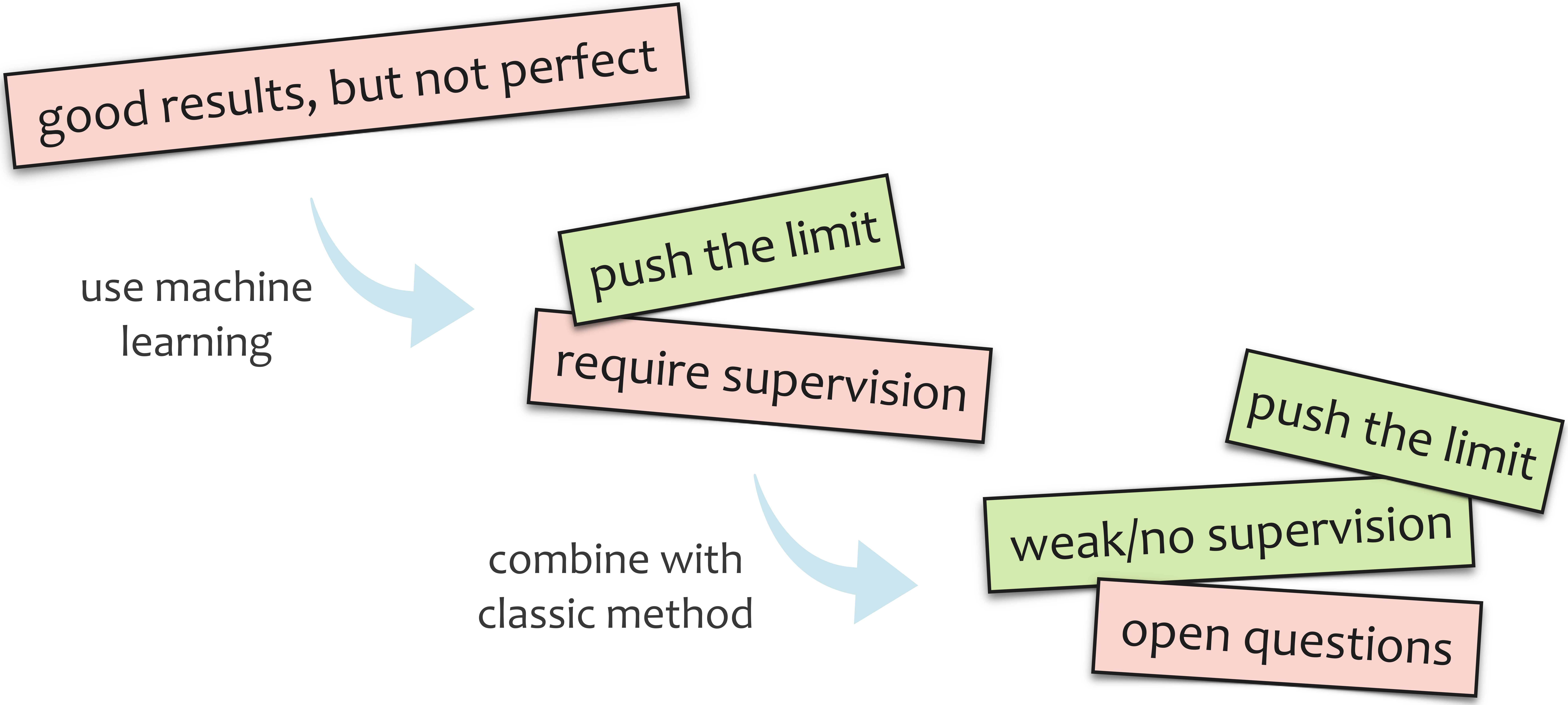
Roufosse et al. 2019

Comparable Results



Roufosse et al. 2019

Key Takeaways



good results, but not perfect

use machine learning

push the limit

require supervision

combine with classic method

push the limit

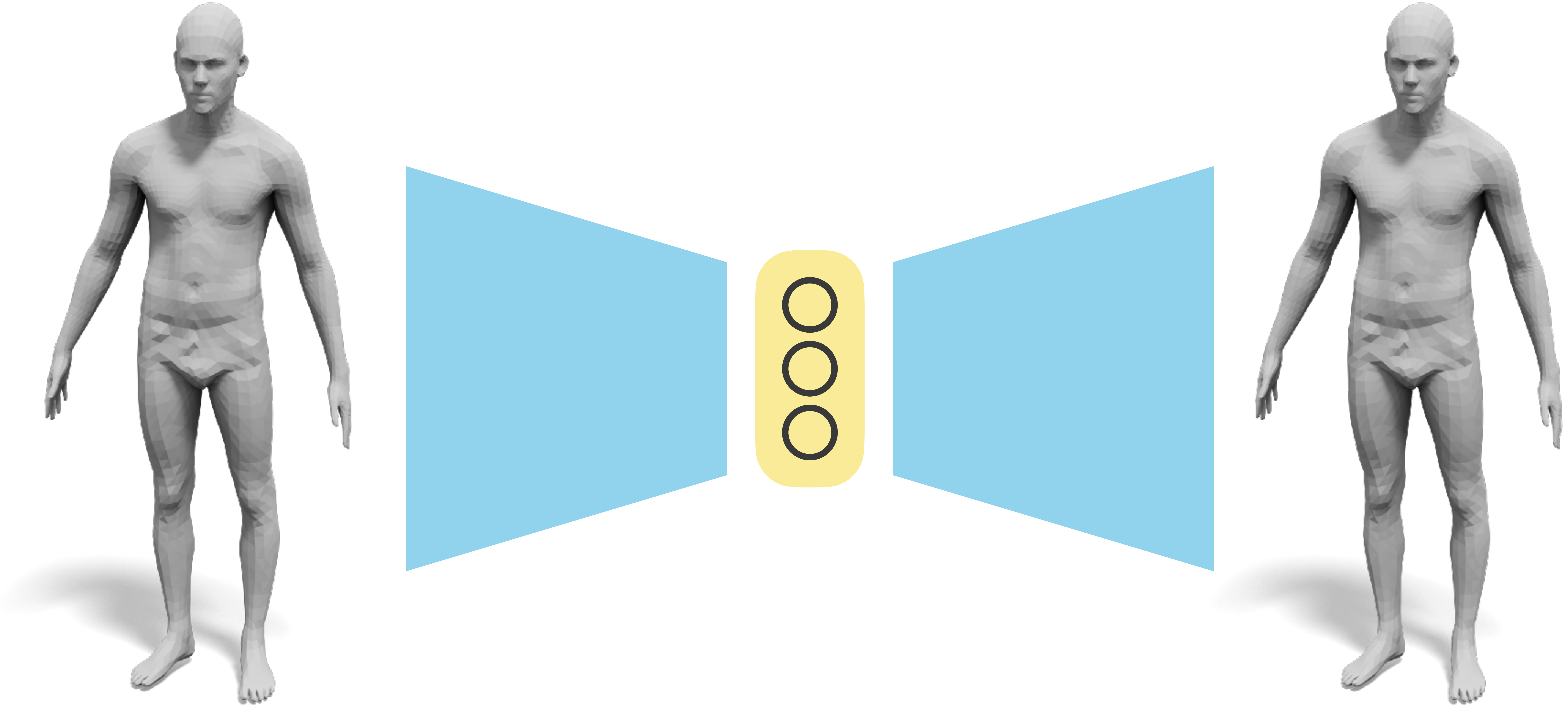
weak/no supervision

open questions

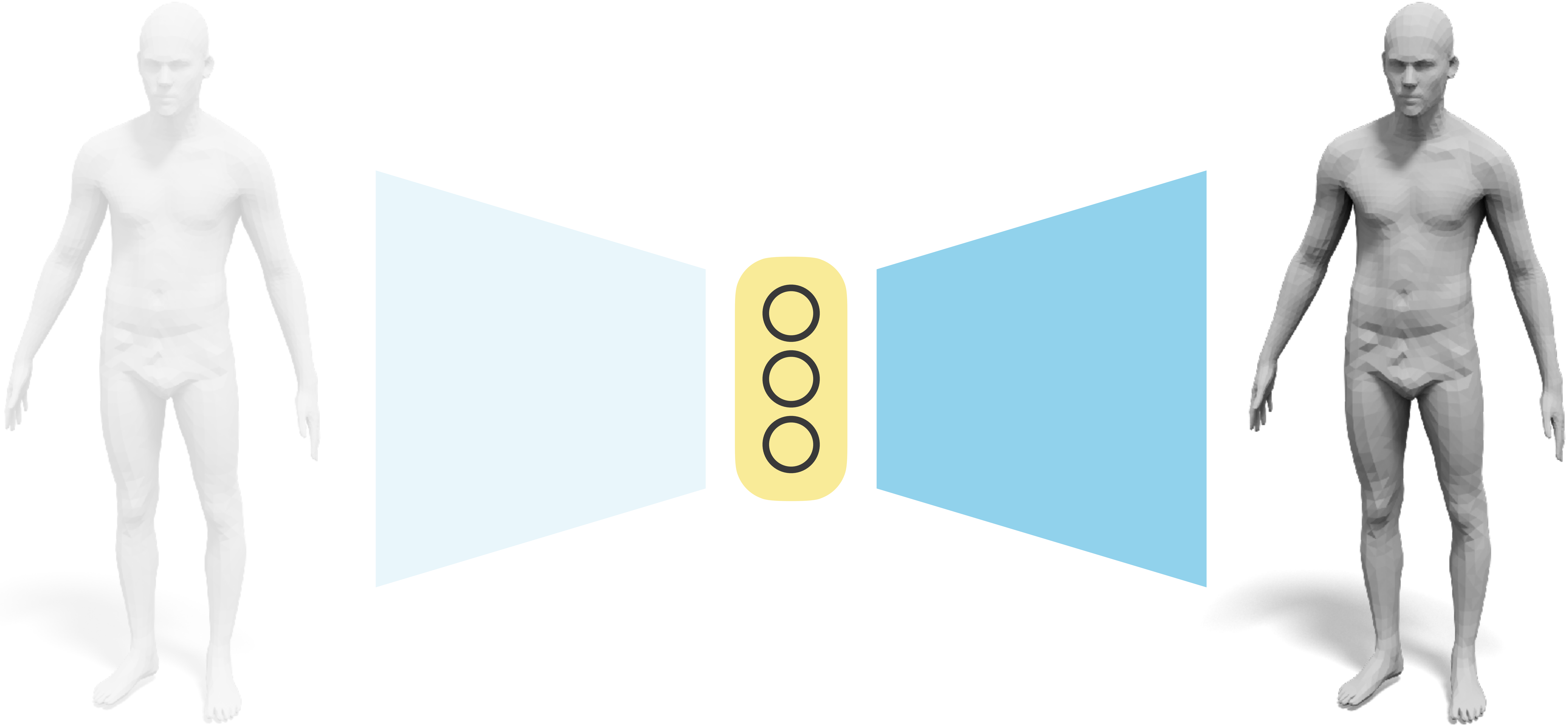
Example: Shape Deformation

image source : Jacobson & Sorkine 2011

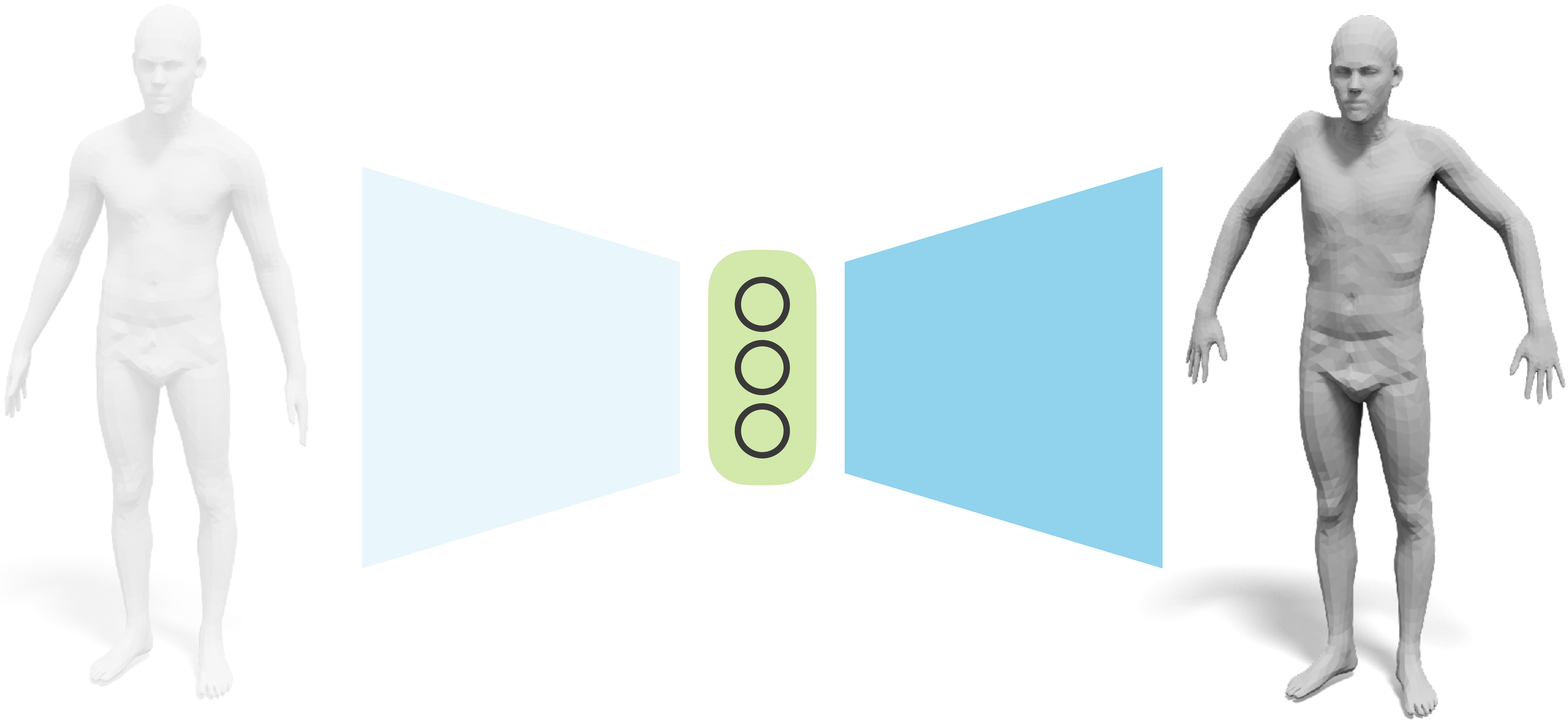
Learning Shape Deformation



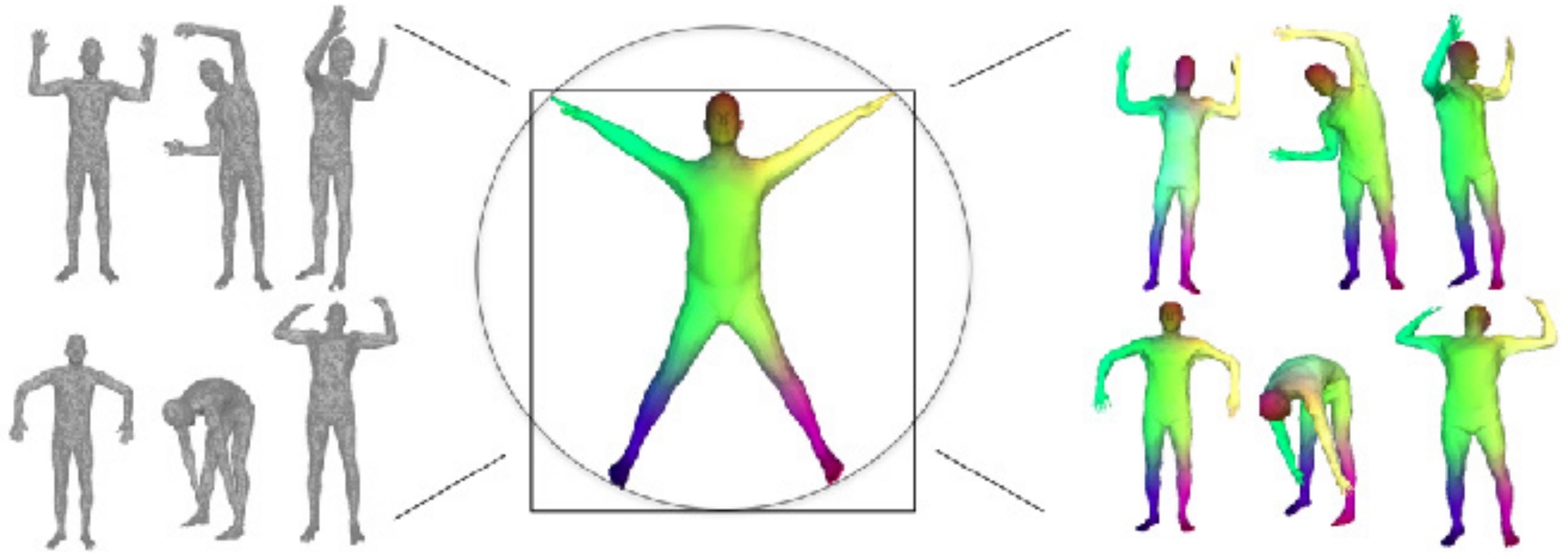
Learning Shape Deformation



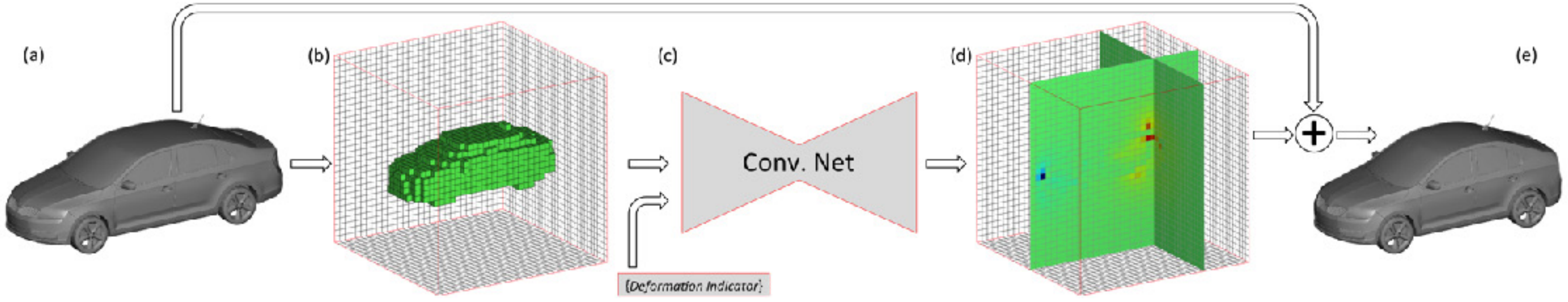
Learning Shape Deformation



Learning Shape Deformation



Groueix et al. 2018



Yumer & Mitra 2016

Gao et al. 2019

Ours
PointNet AE

Rakotosaona & Ovsjanikov, 2020

Issues of preserving details

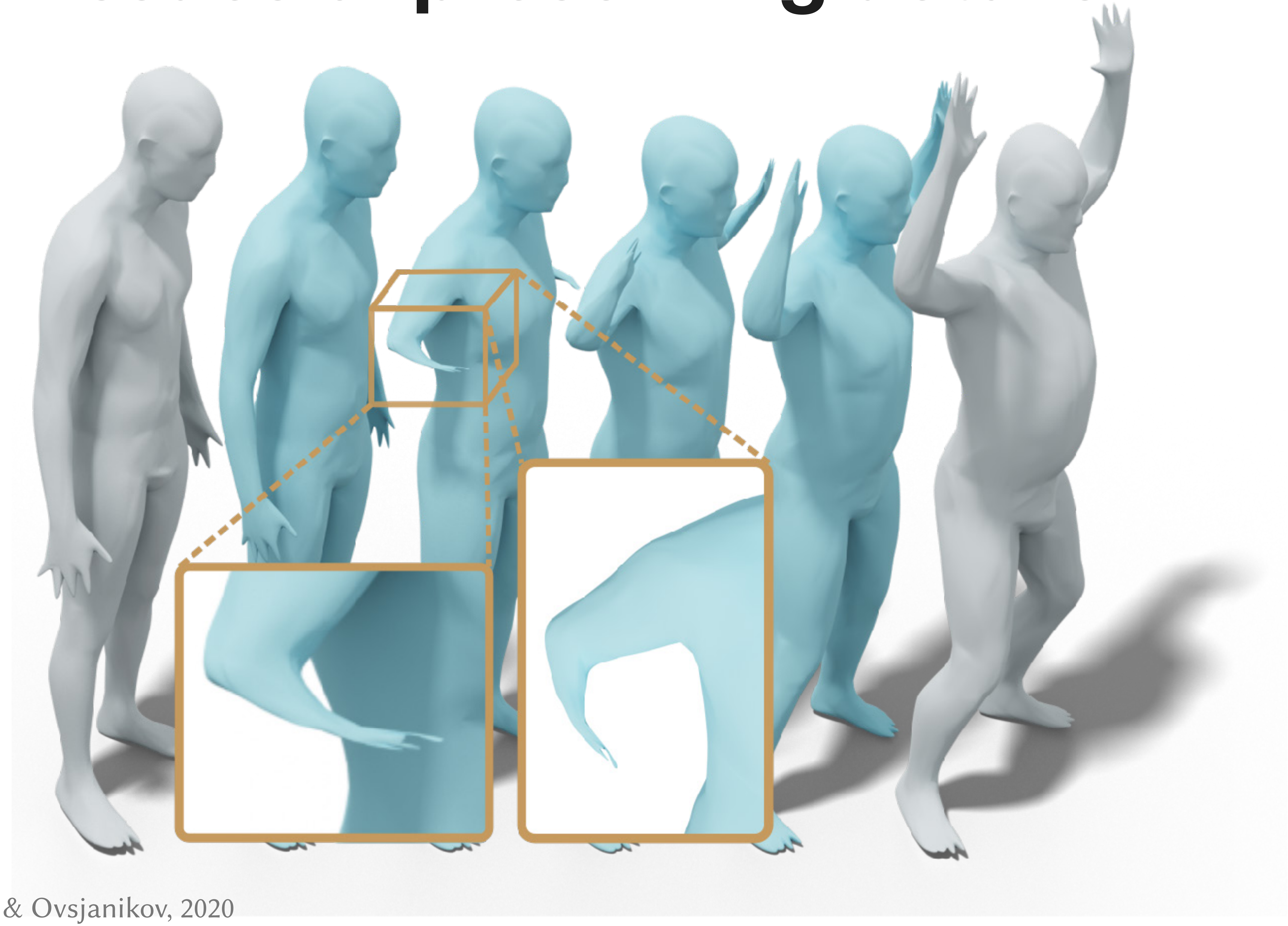
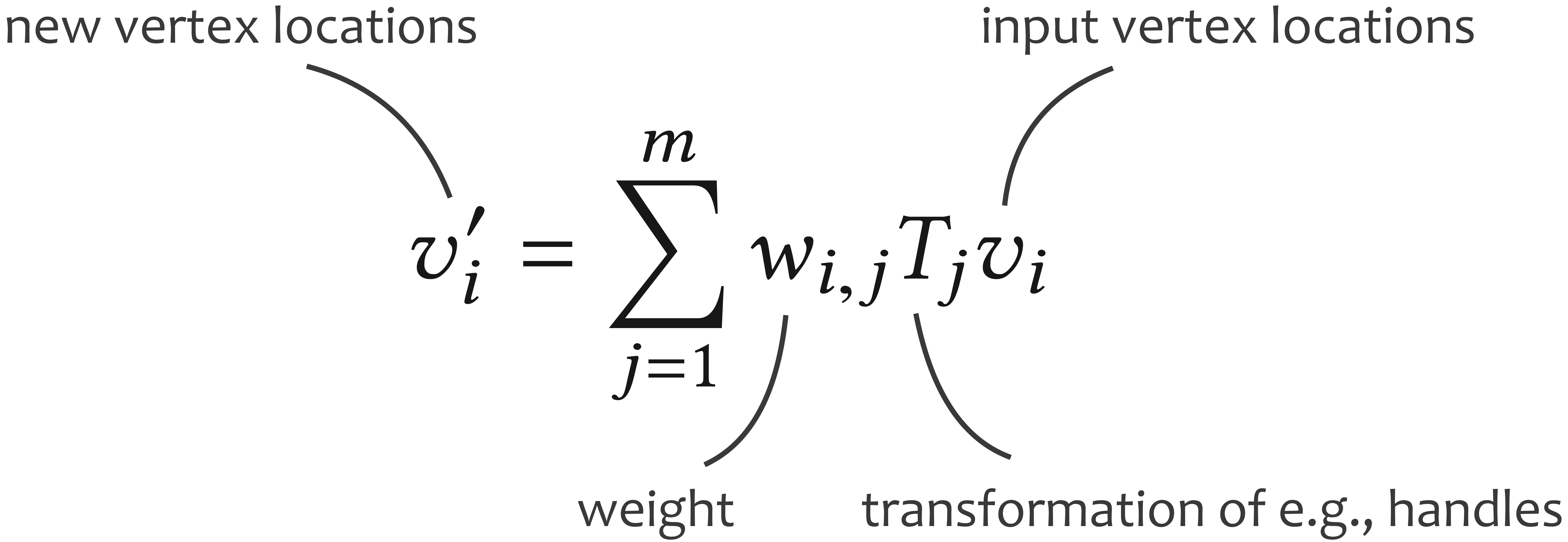


image source: Rakotosaona & Ovsjanikov, 2020

Linear Blend Skinning



$$v_i' = \sum_{j=1}^{m} w_{i,j} T_j v_i$$

new vertex locations

input vertex locations

weight

transformation of e.g., handles

Linear Blend Skinning

input

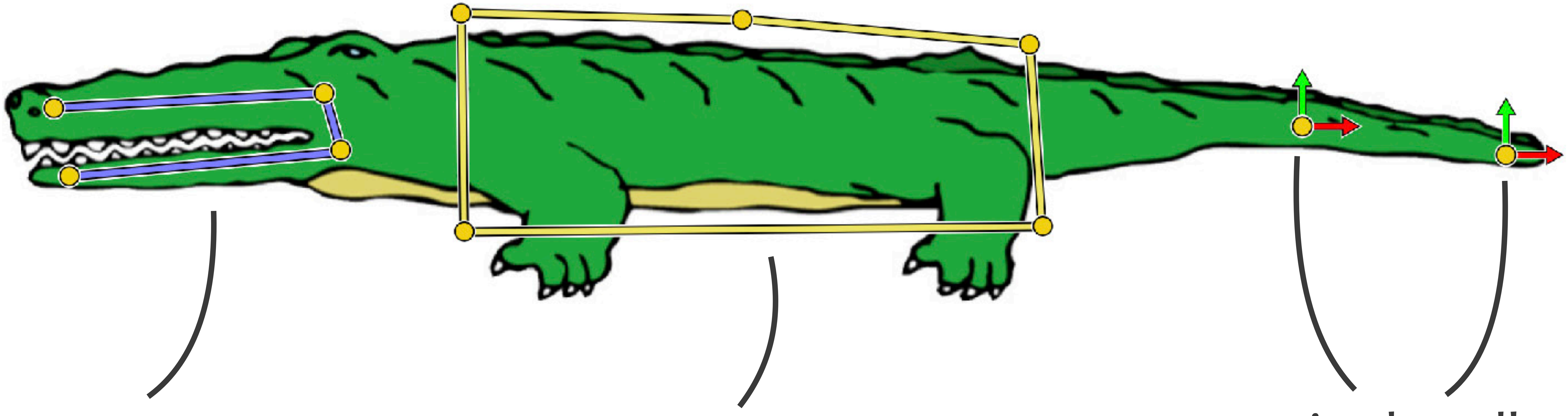
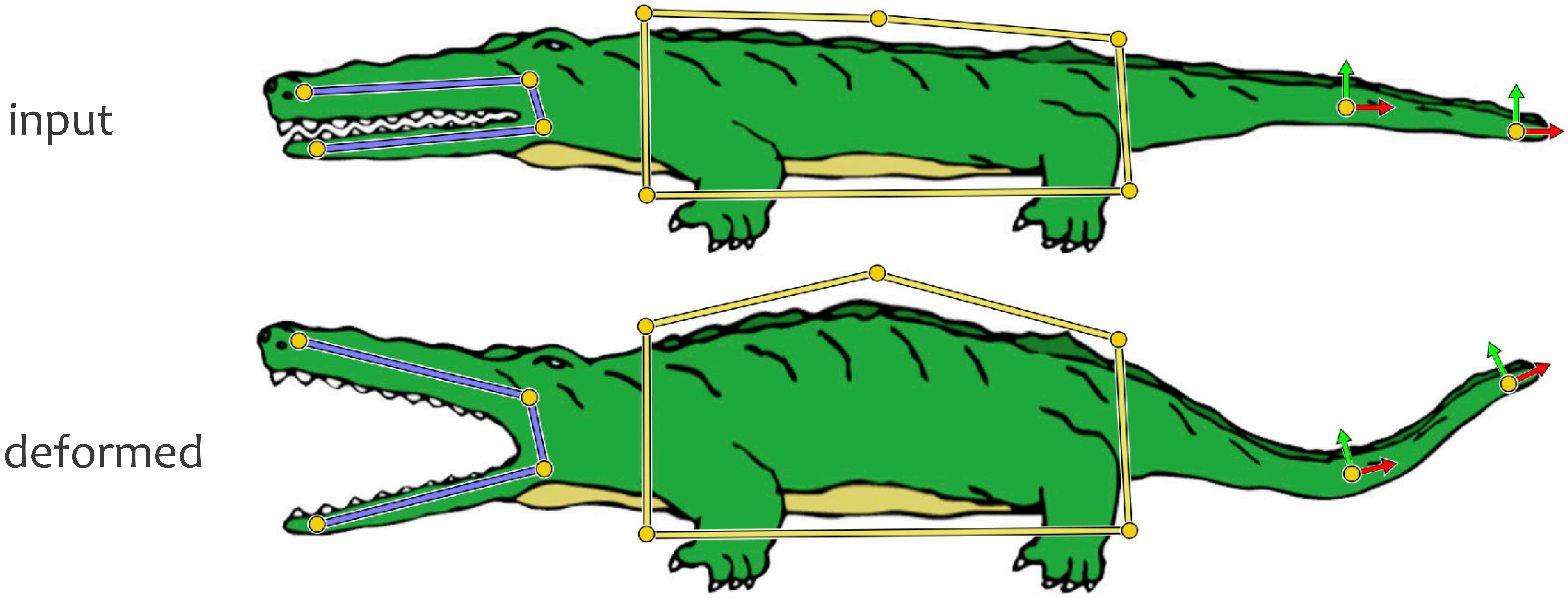


image source: Jacobson et al. 2011

Linear Blend Skinning



input

deformed

image source: Jacobson et al. 2011

Linear Blend Skinning

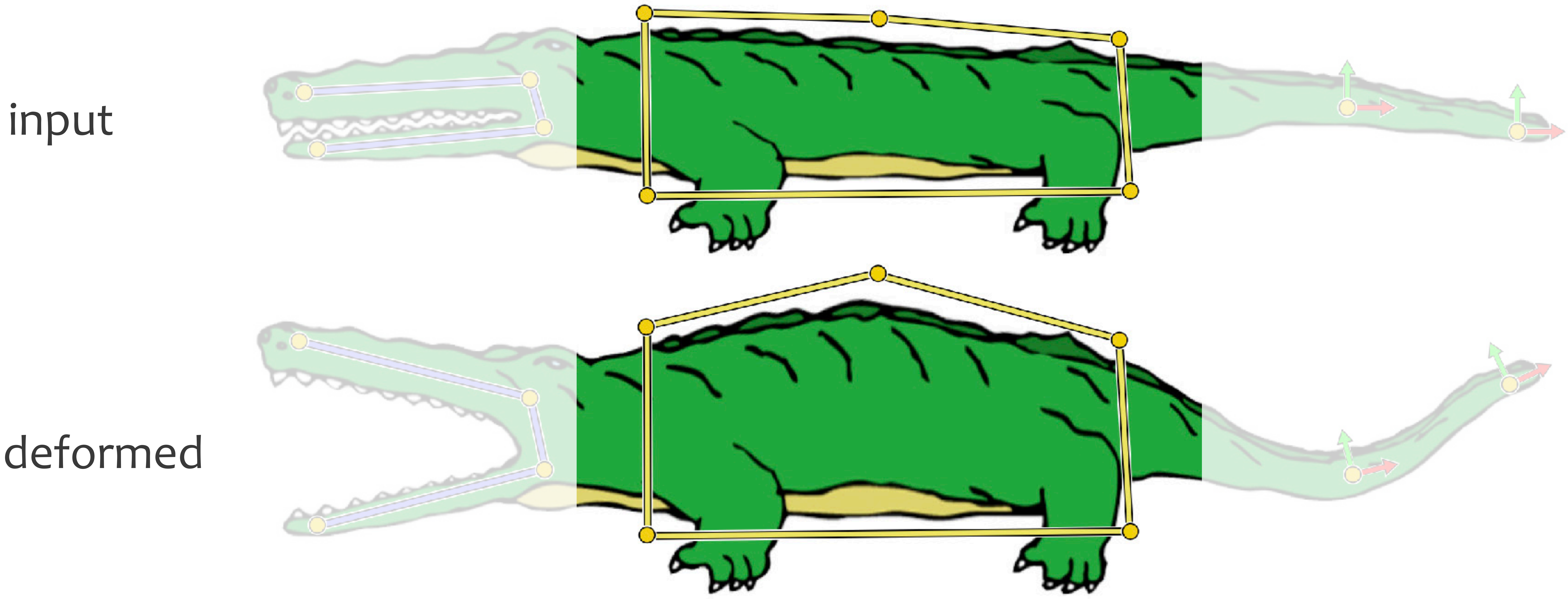
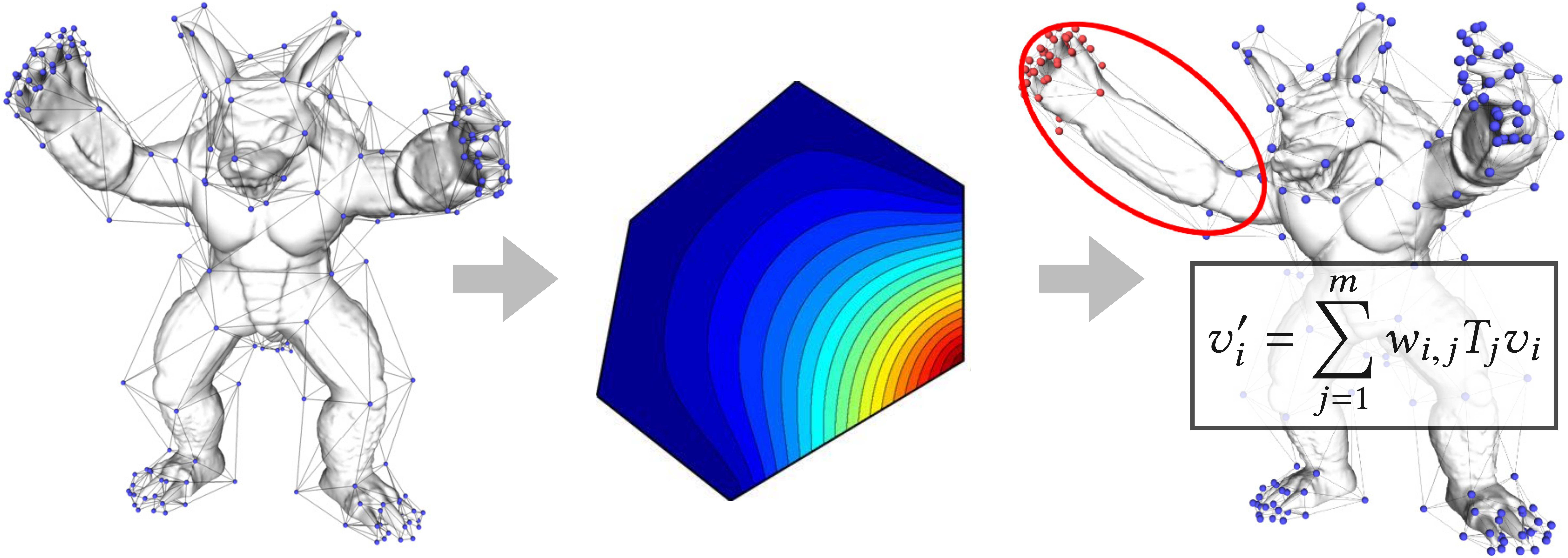


image source: Jacobson et al. 2011

Cage-Based Deformation



$$v_i' = \sum_{j=1}^{m} w_{i,j} T_j v_i$$

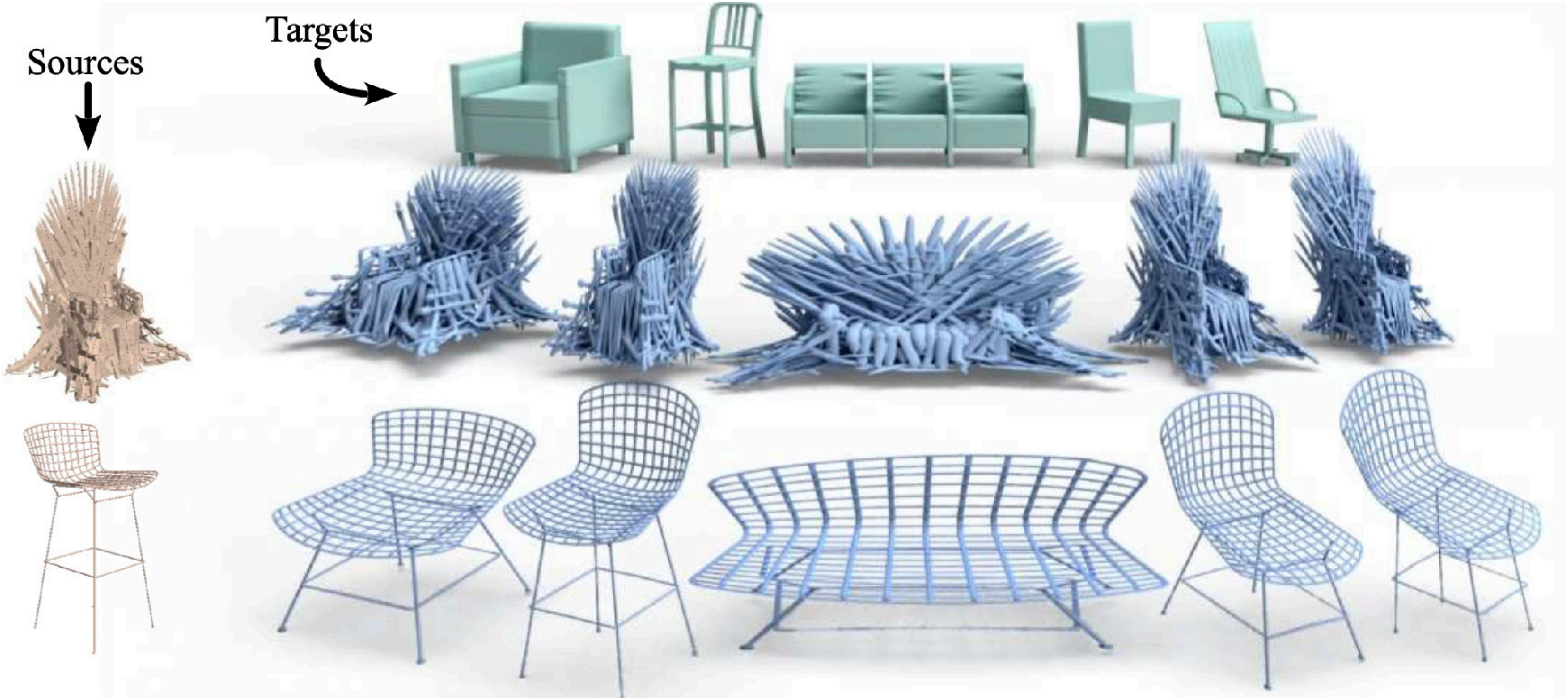
construct a cage

compute cage weights

deform the model

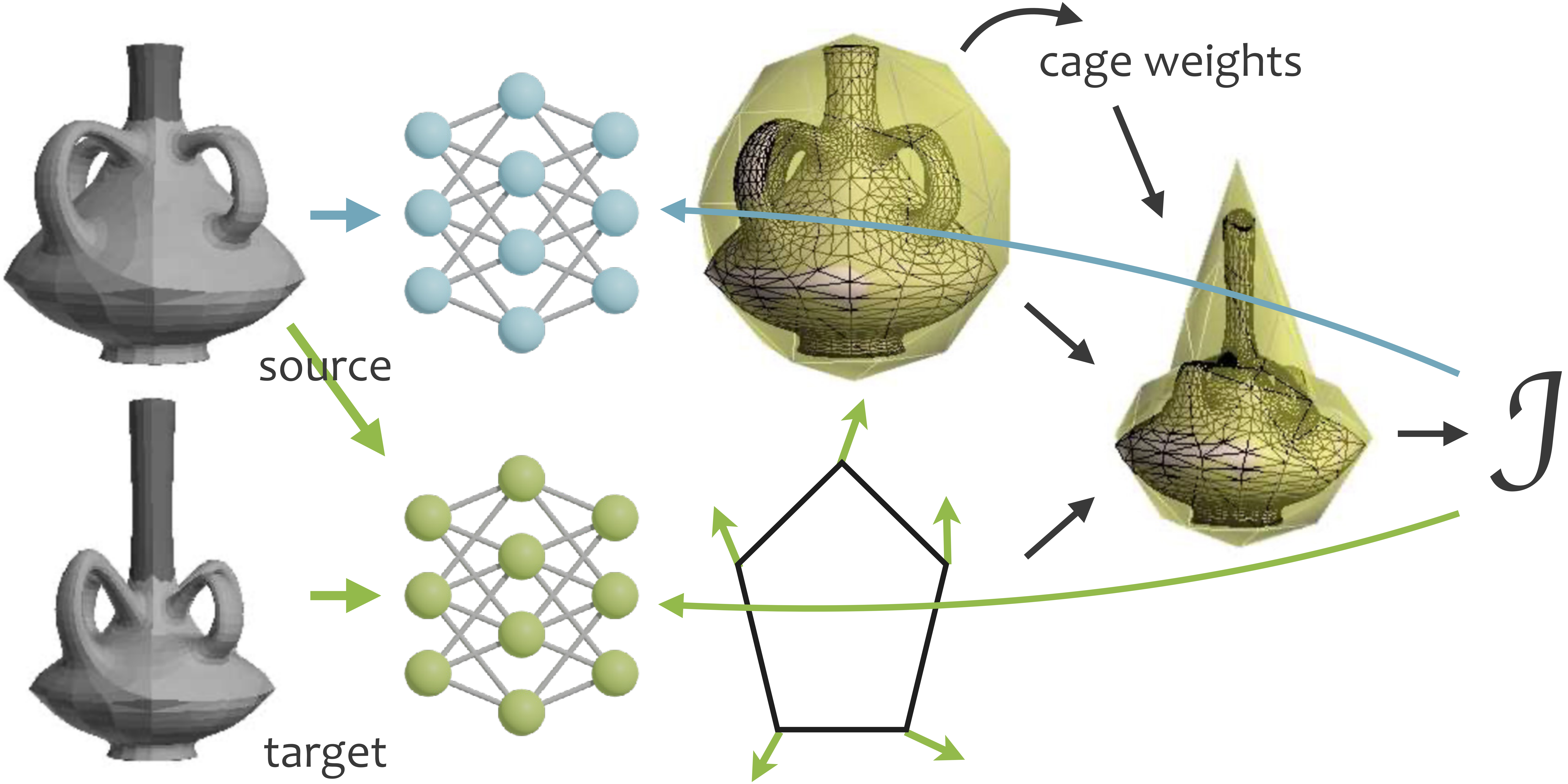
image source: Casti et al. 2018, Budninskiy et al. 2016

Neural Cages



Yifan et al. 2020

Neural Cages



Yifan et al. 2020

Loss Function

$$\mathcal{J} = w_1 \mathcal{L}_{\mathrm{MVC}} + w_2 \mathcal{L}_{\mathrm{align}} + w_3 \mathcal{L}_{\mathrm{shape}}$$

“nice” cage
(e.g., not self-overlap)

positive cage weights

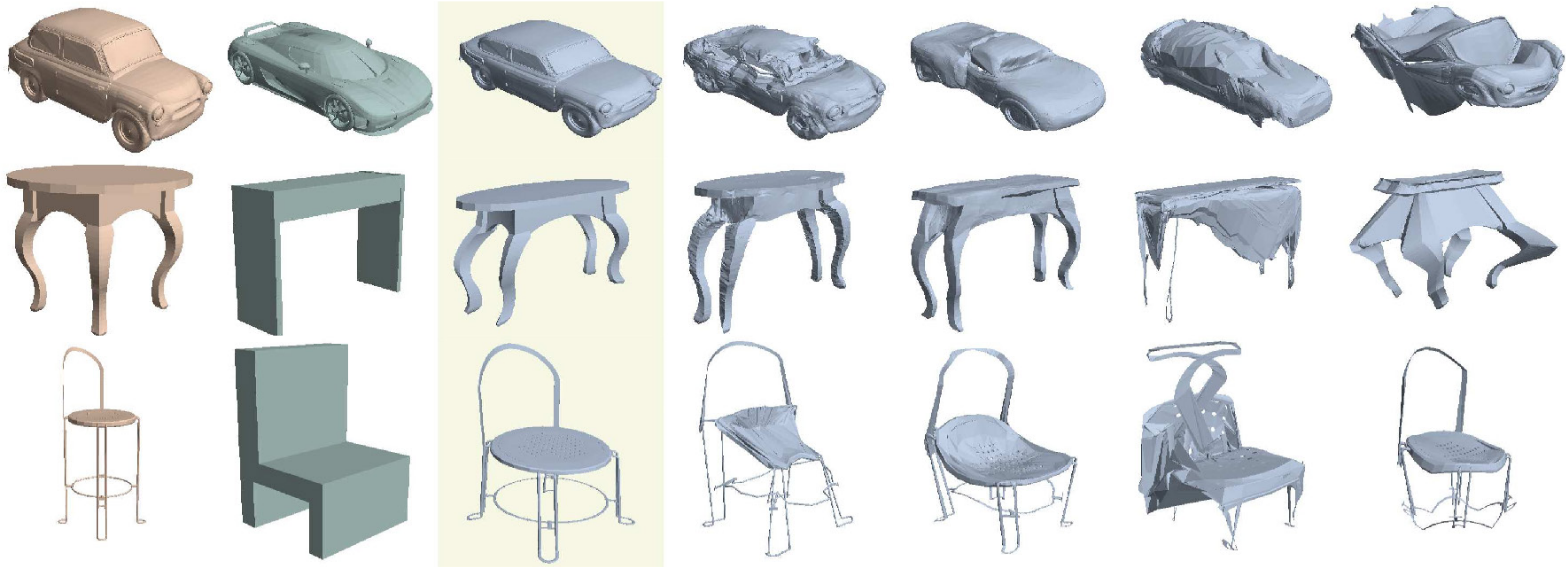
align with target

Chamfer

preserve input
shapes

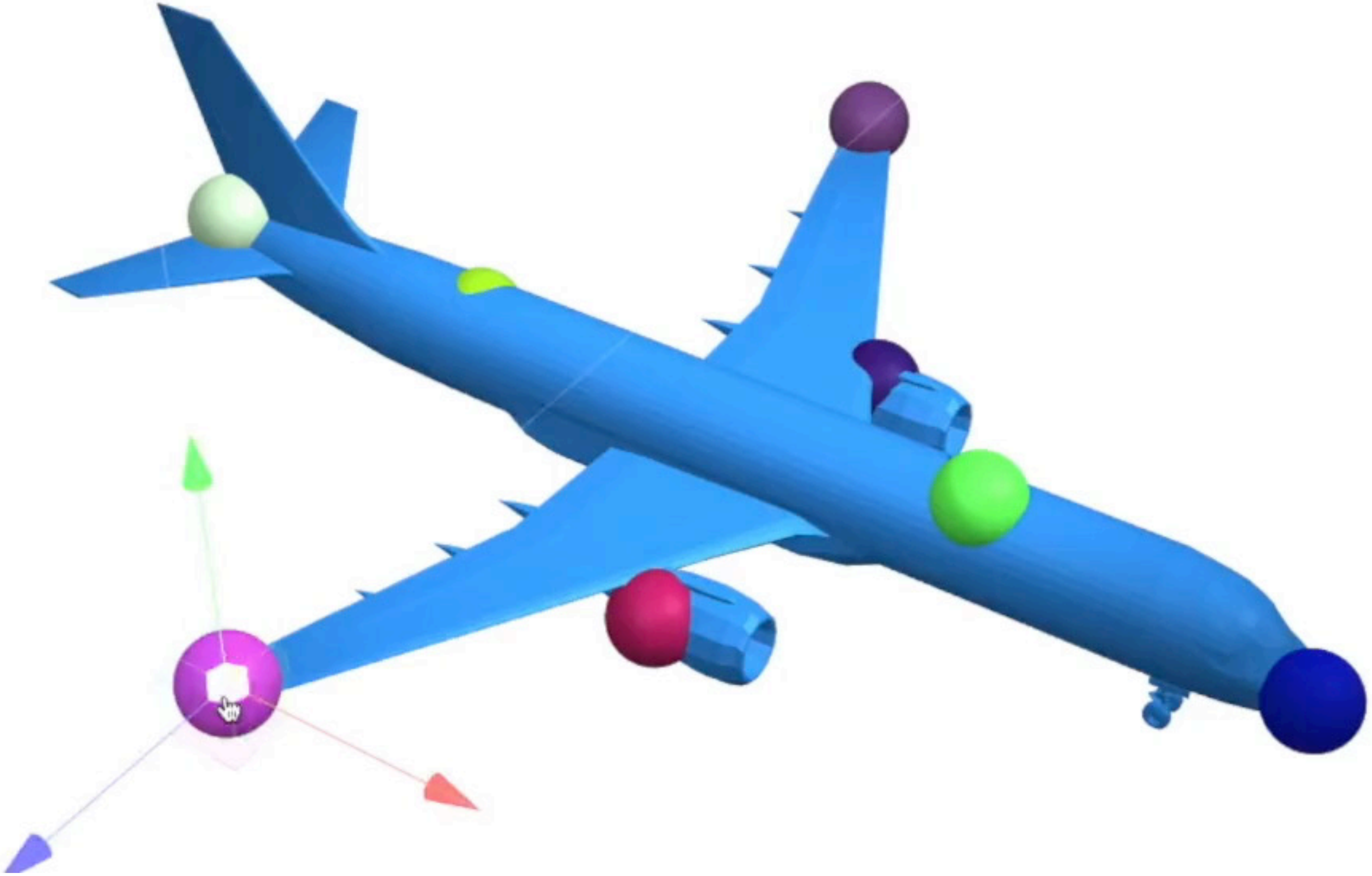
e.g., preserve normals

Detail-Preserving Deformation



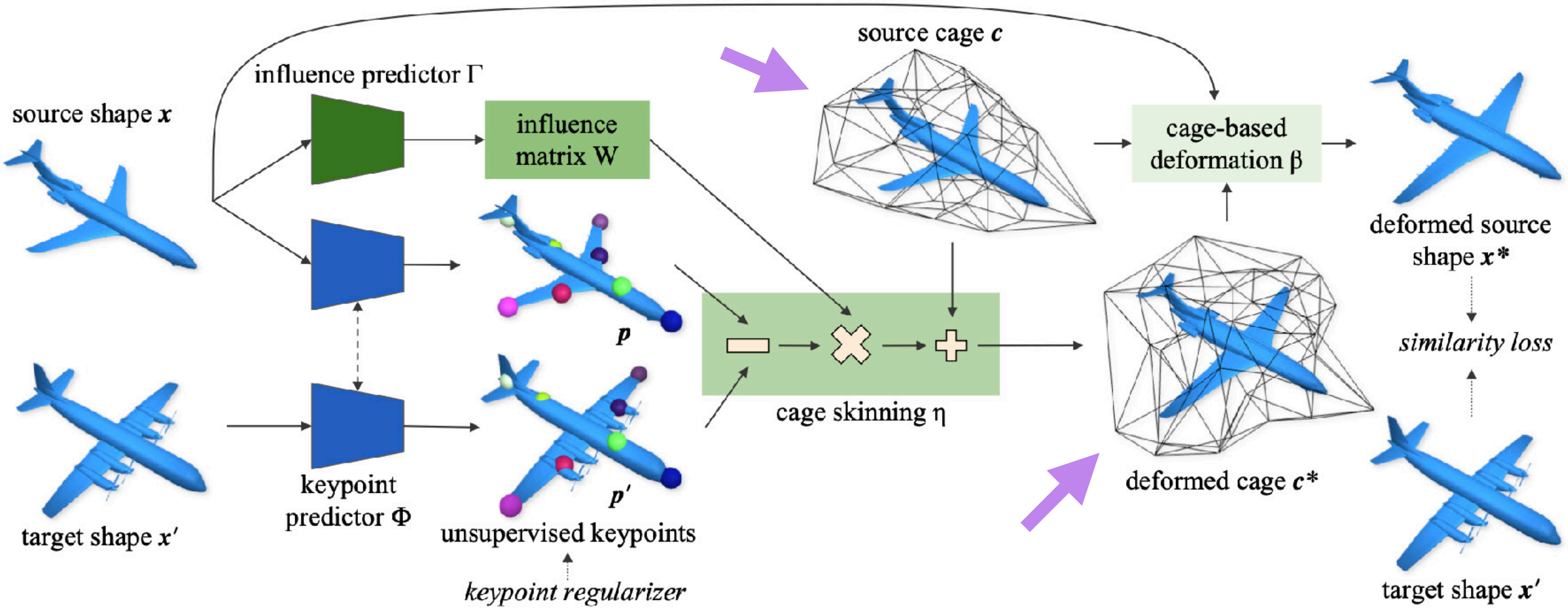
input target output previous methods

Keypoint Deformer



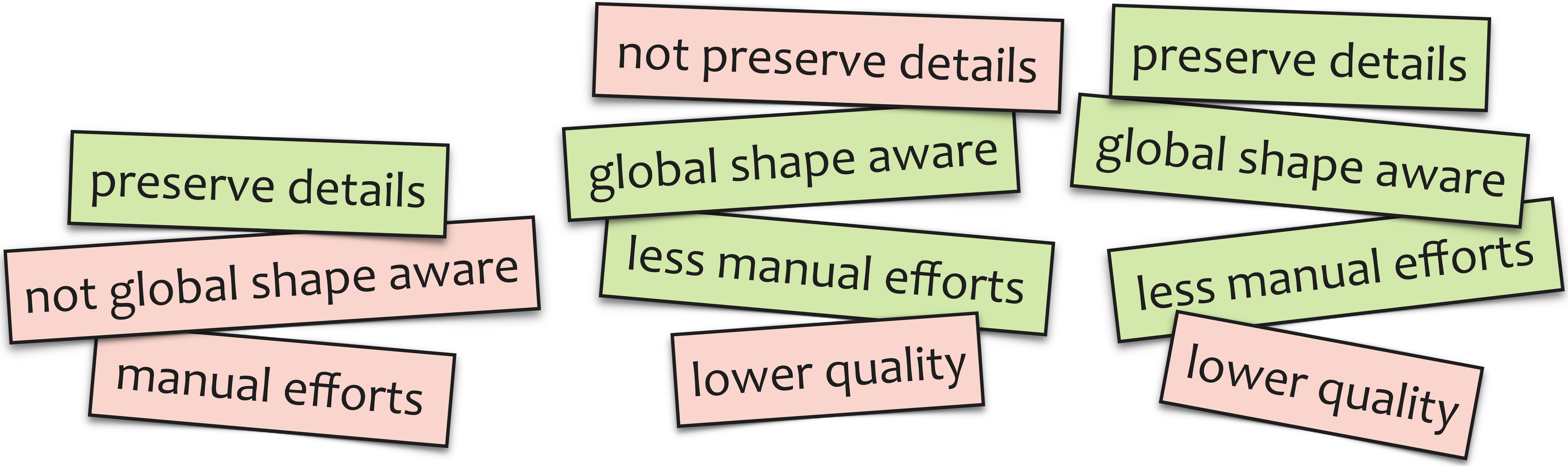
Jakab et al. 2021

Keypoint Deformer



Jakab et al. 2021

Key Takeaways



preserve details
not global shape aware
manual efforts

Classic

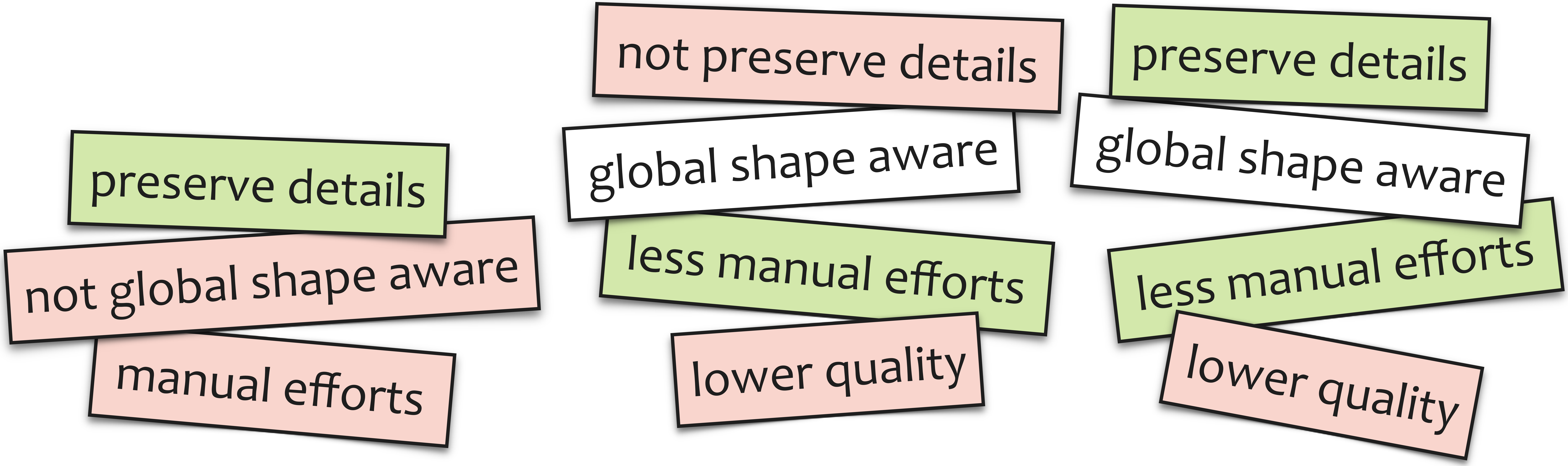
not preserve details
global shape aware
less manual efforts
lower quality

Deep

preserve details
global shape aware
less manual efforts
lower quality

Classic + Deep

Key Takeaways



preserve details
not global shape aware
manual efforts

Classic

not preserve details
global shape aware
less manual efforts
lower quality

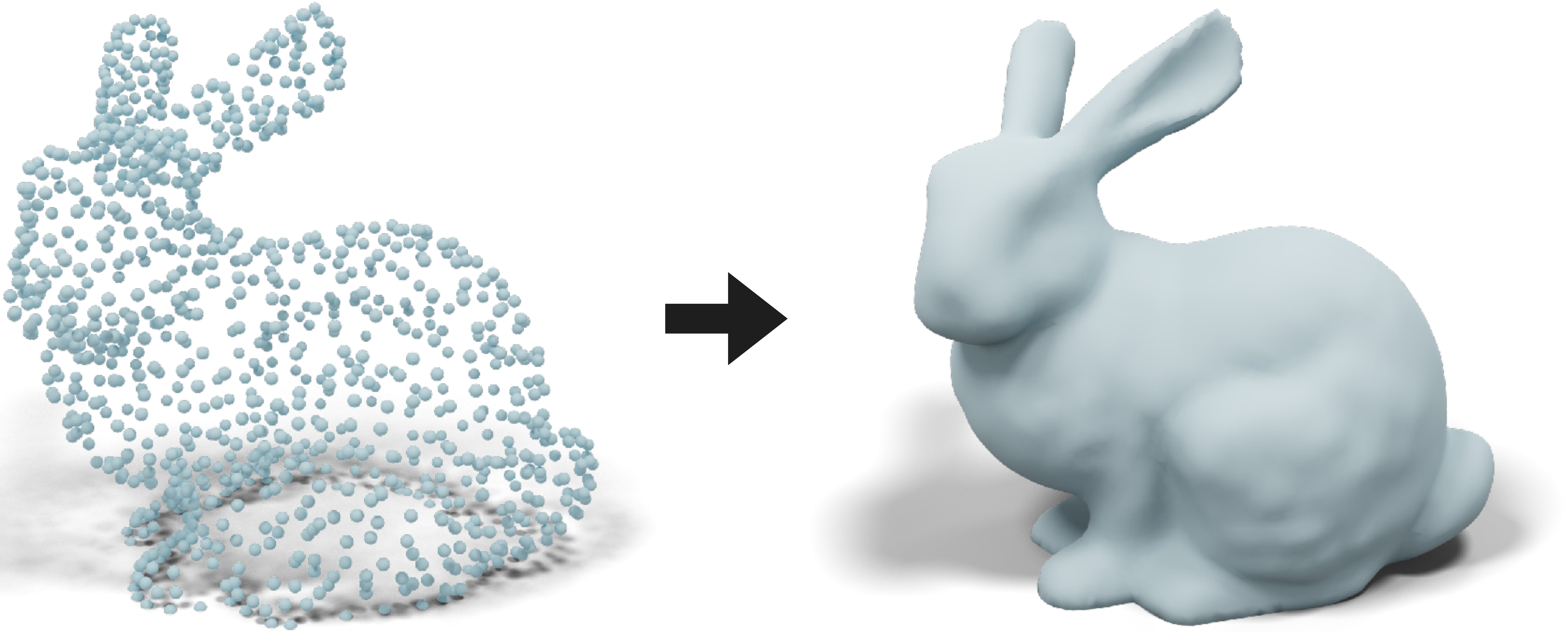
Deep

preserve details
global shape aware
less manual efforts
lower quality

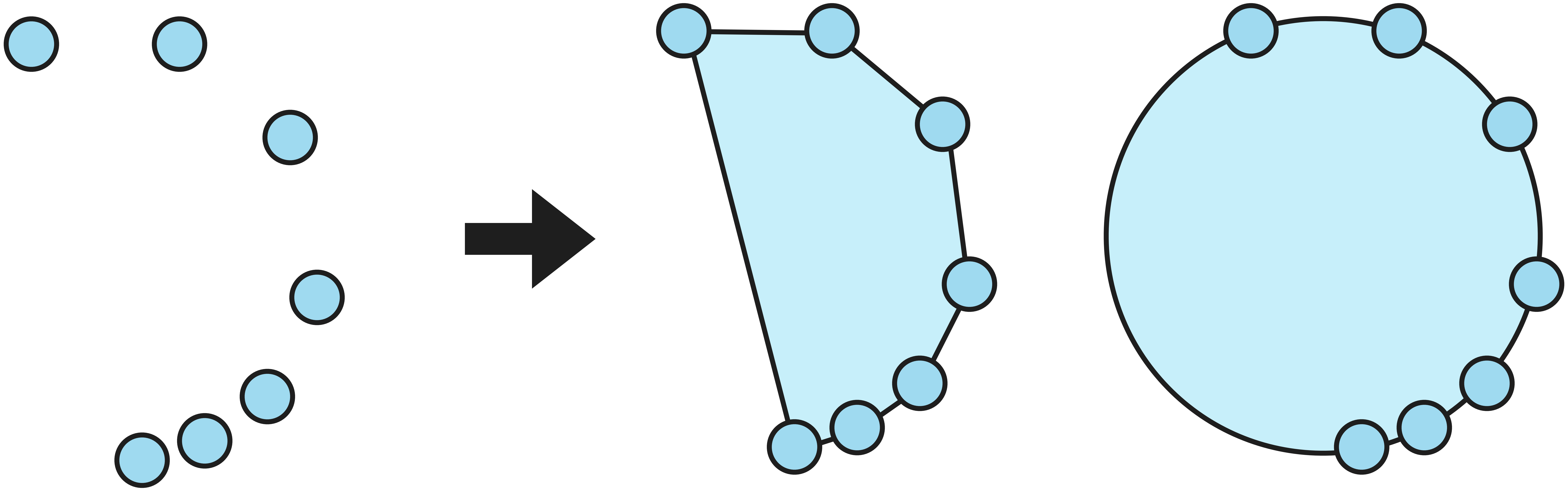
Classic + Deep

Machine learning as geometric prior

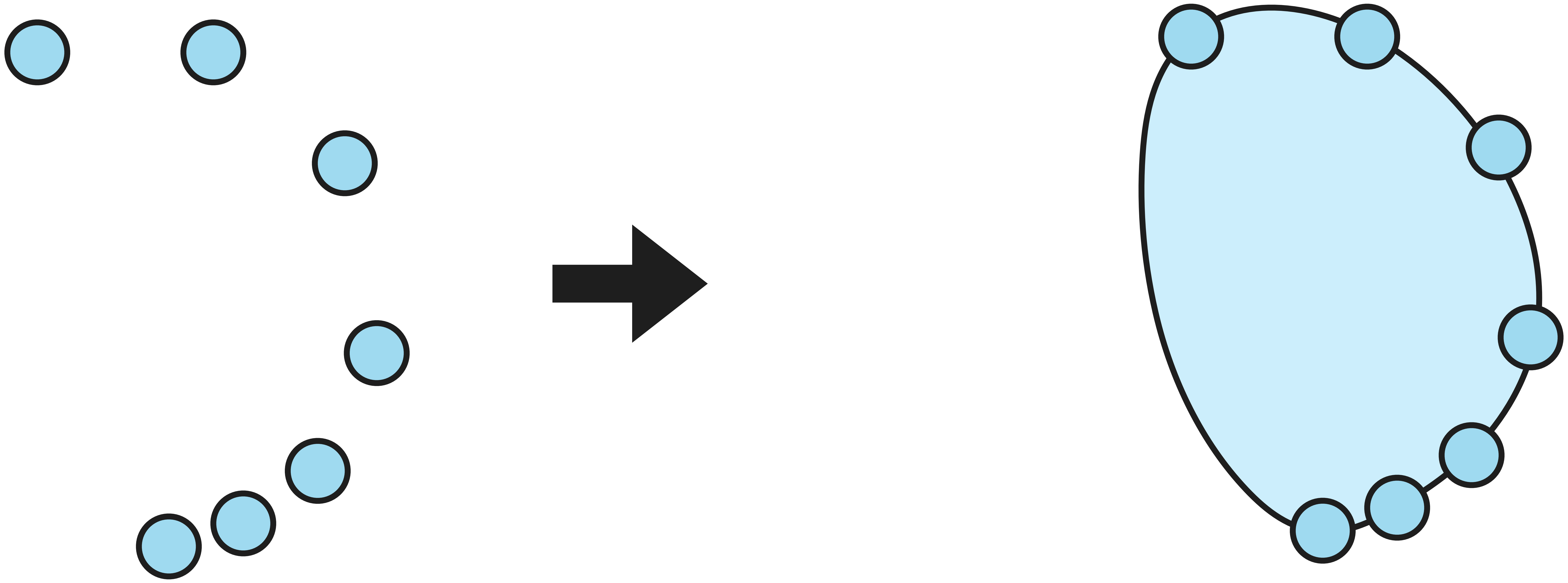
Surface Reconstruction



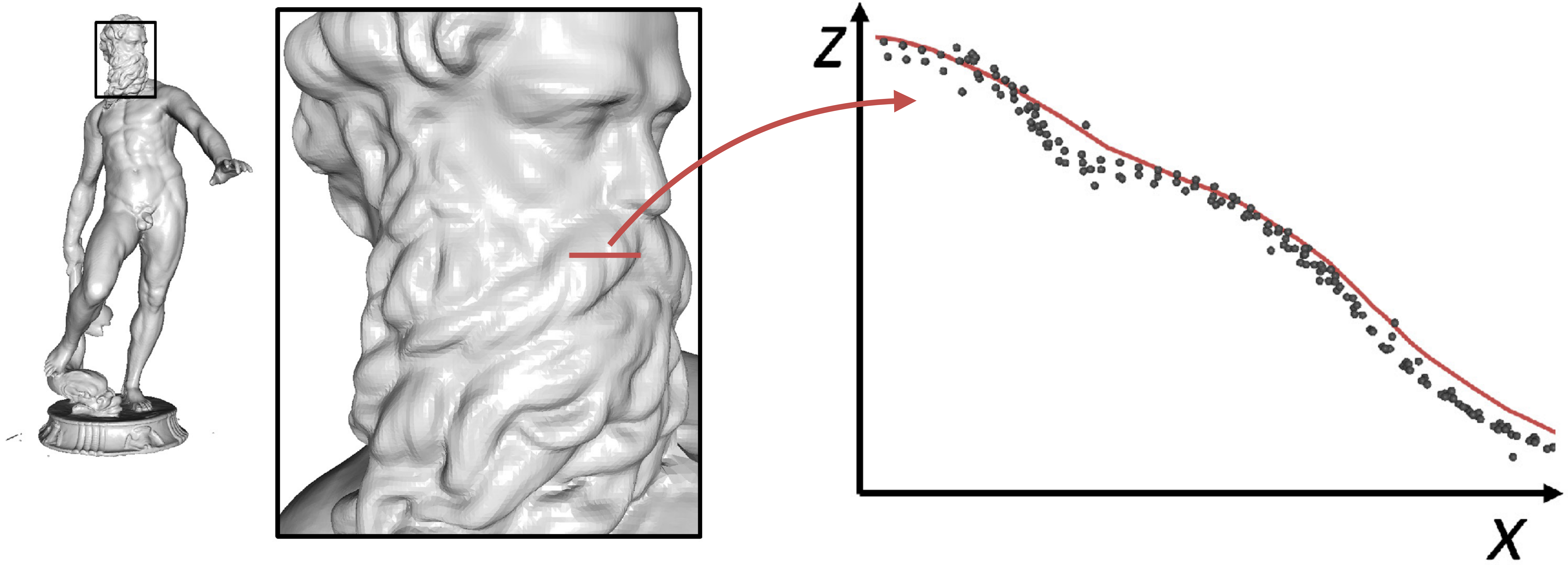
An ill-posed problem



A Smooth Solution

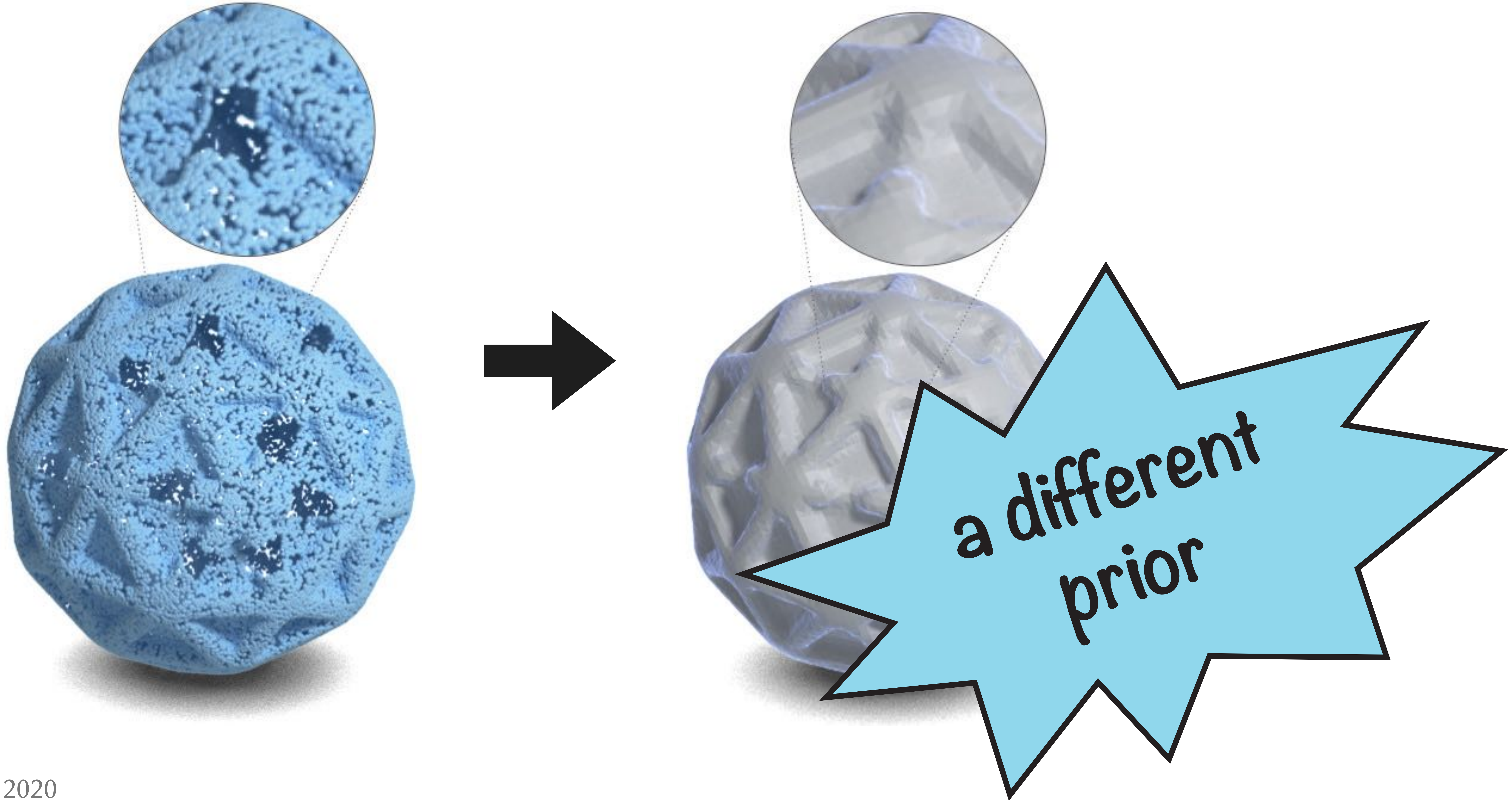


Classic Smoothness Prior



Kazhdan et al. 2006

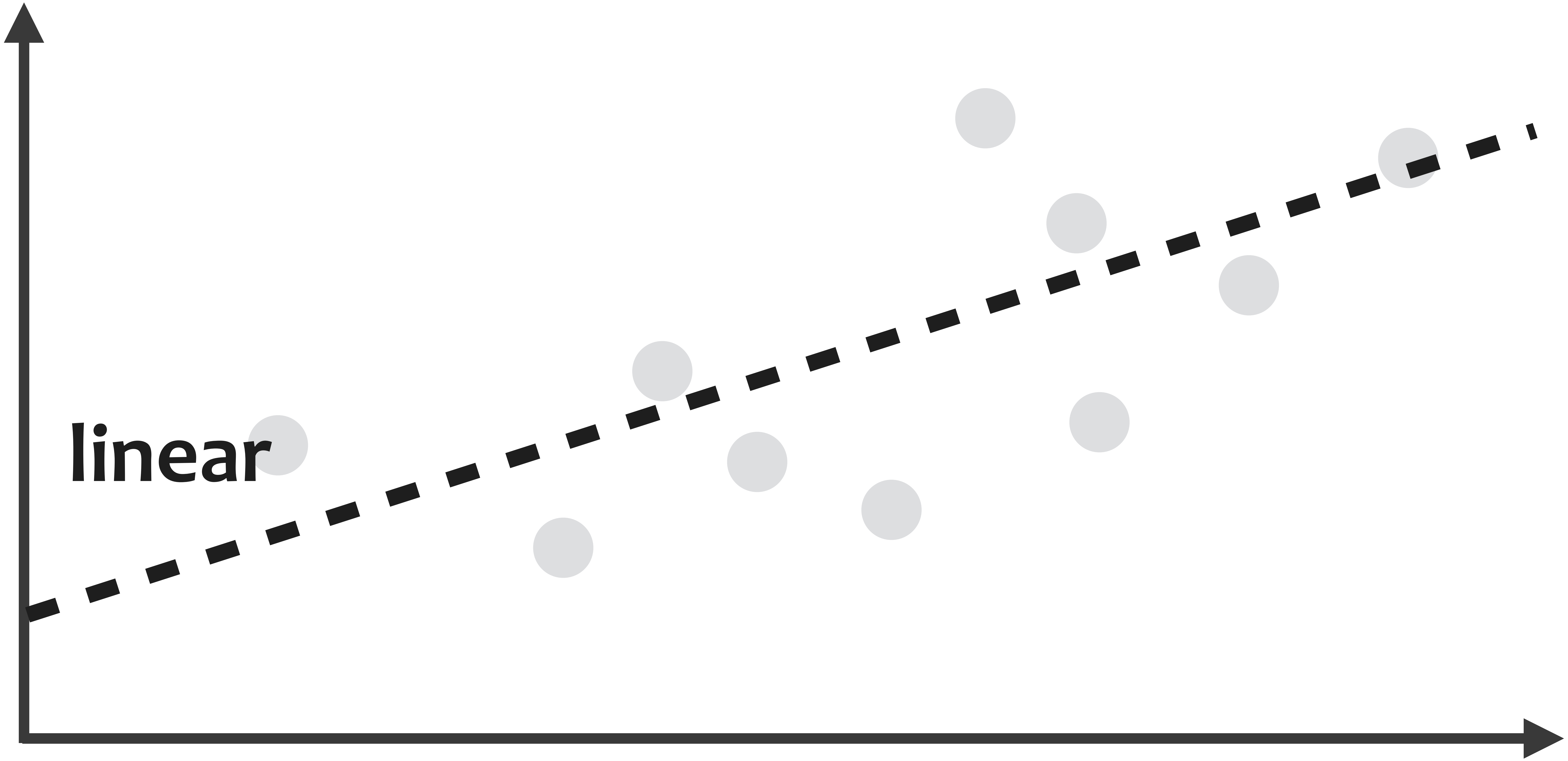
Smoothness is not always good



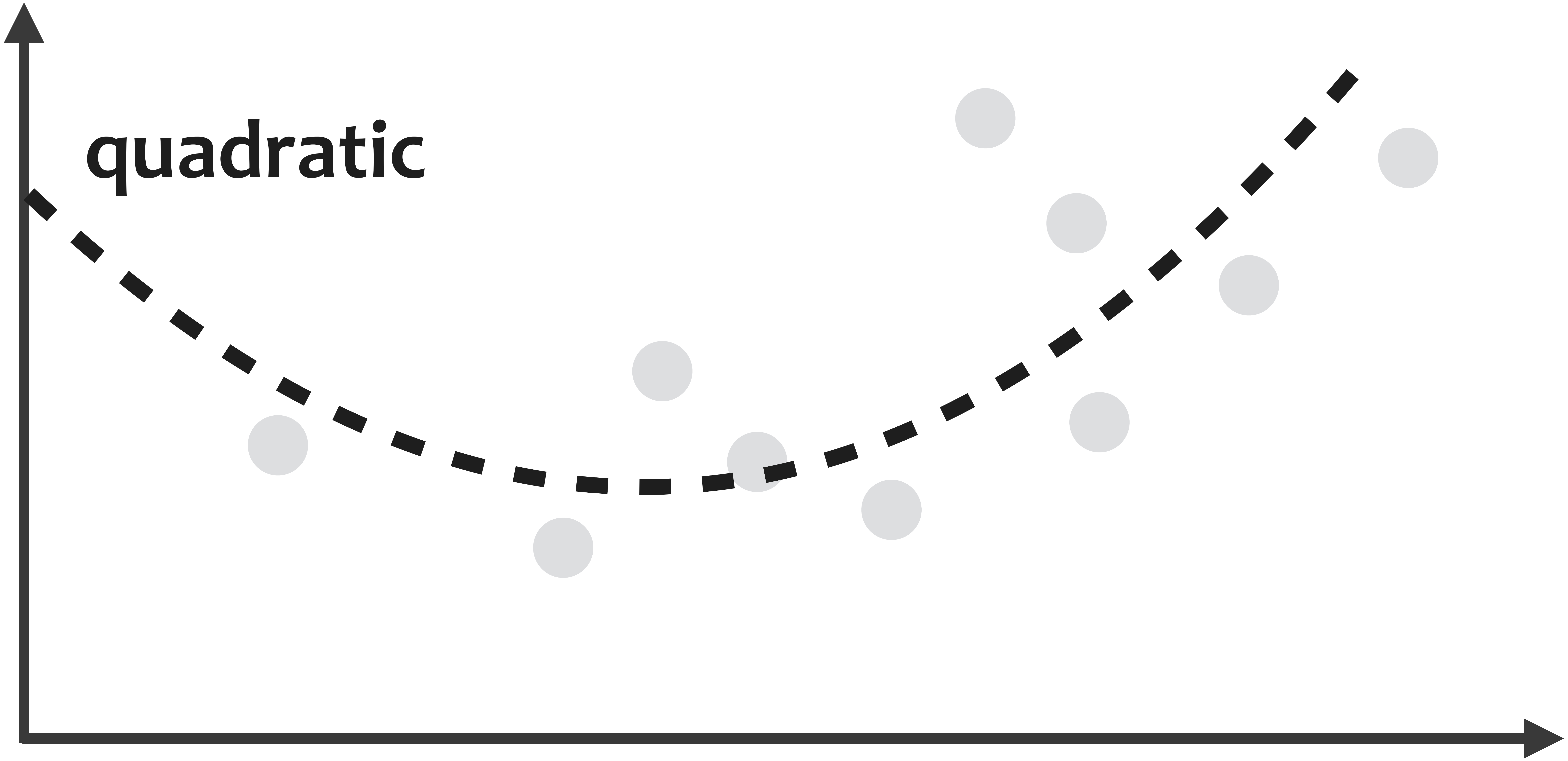
a different prior

image source: Hanocka et al. 2020

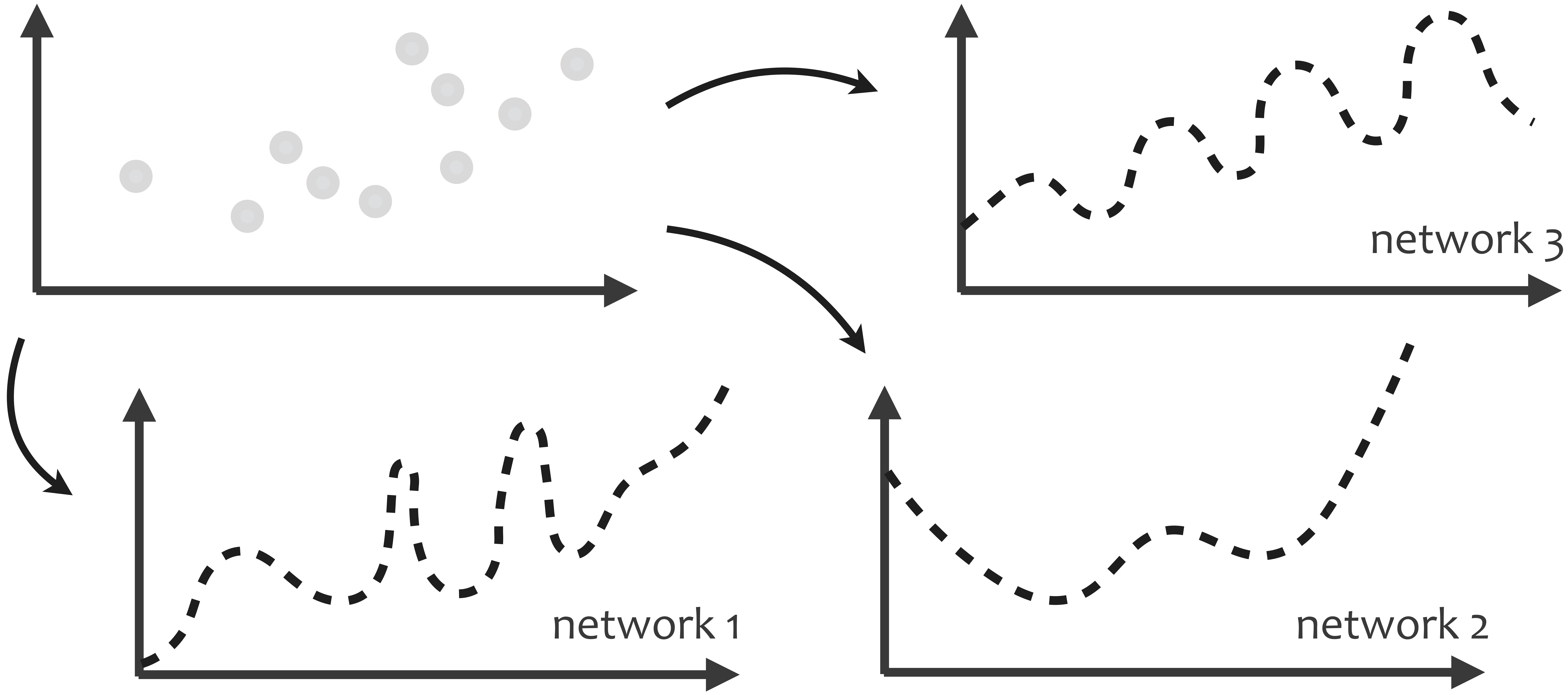
Inductive Bias as Prior



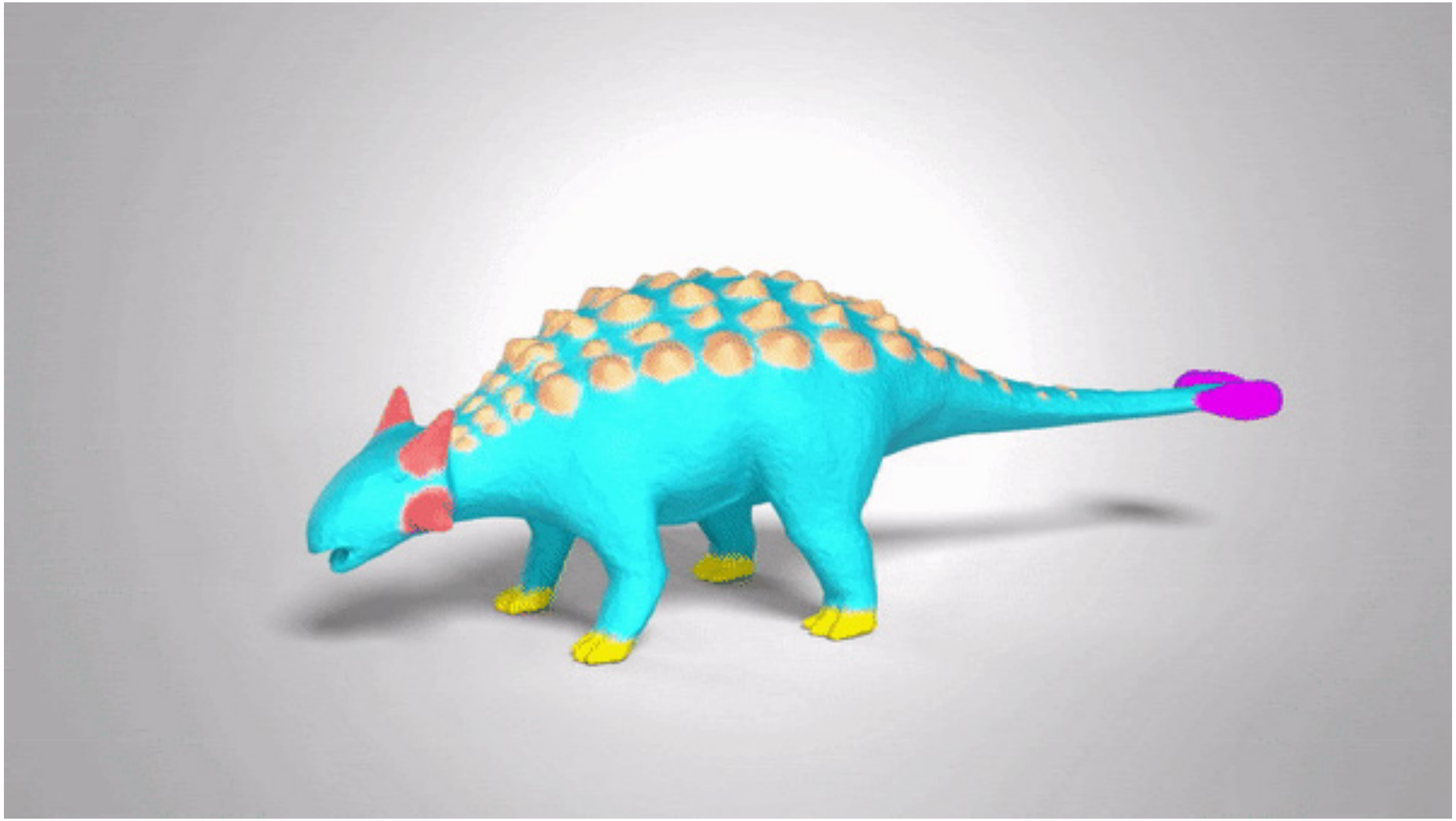
Inductive Bias as Prior



Deep Network Biases

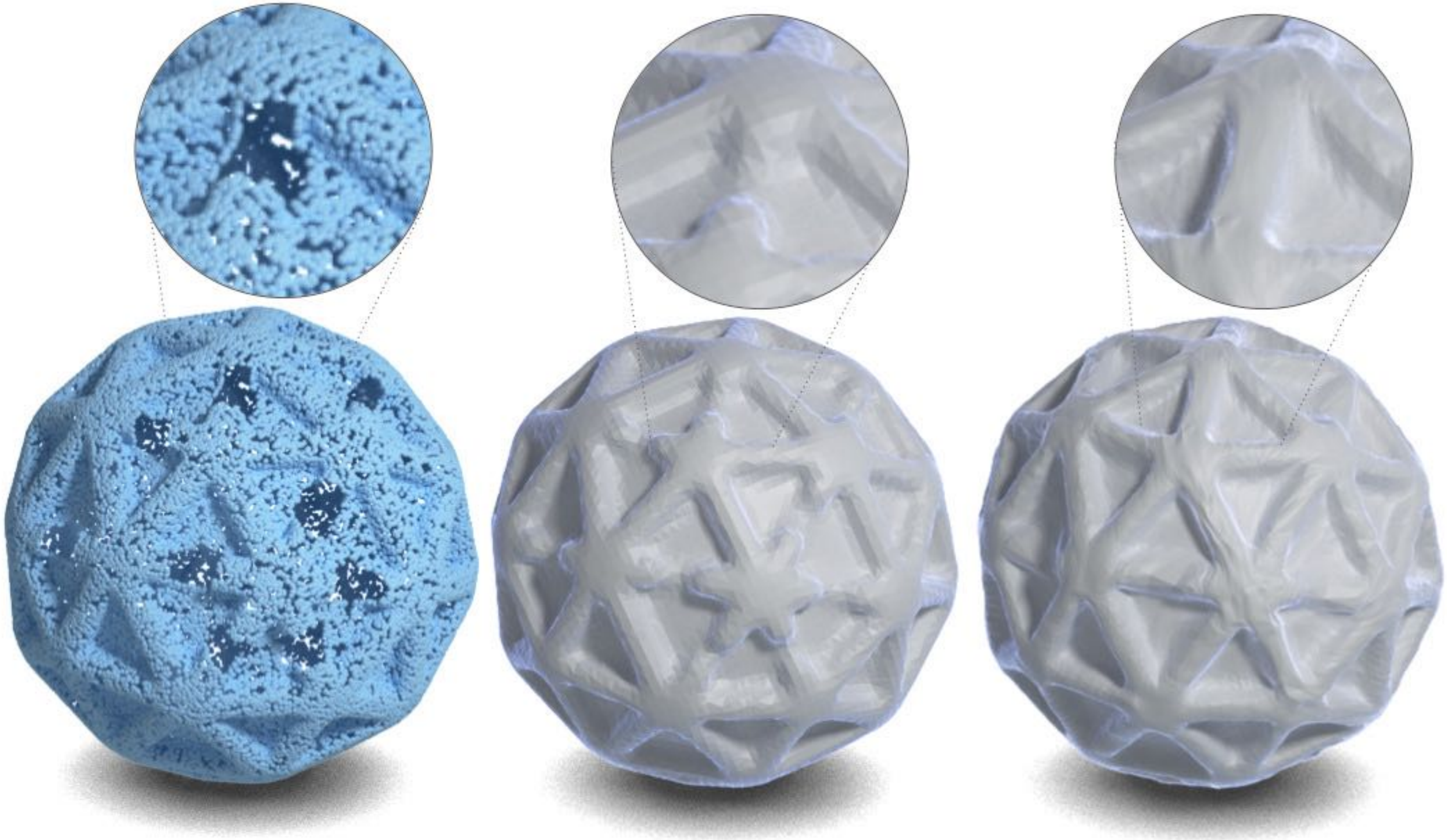


New Possibilities : Self-Prior



Hanocka et al. 2020

Results of Self-Prior



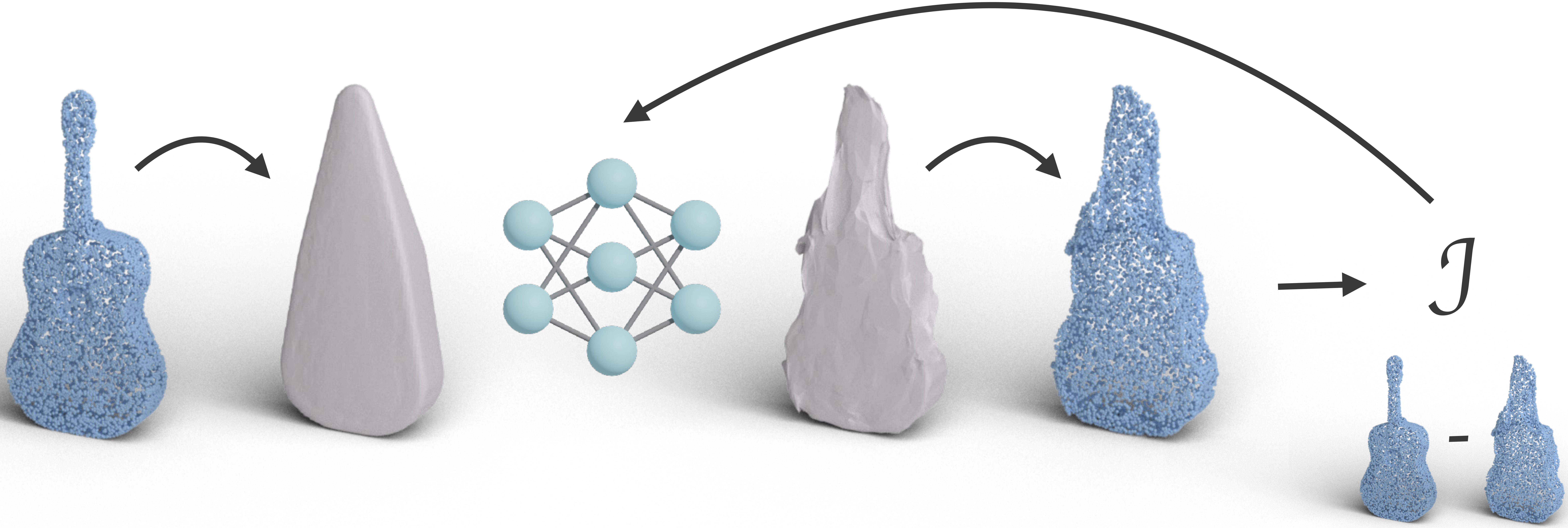
input

Smoothness
[Kazhdan et al. 2006]

Self-prior
[Hanocka et al. 2020]

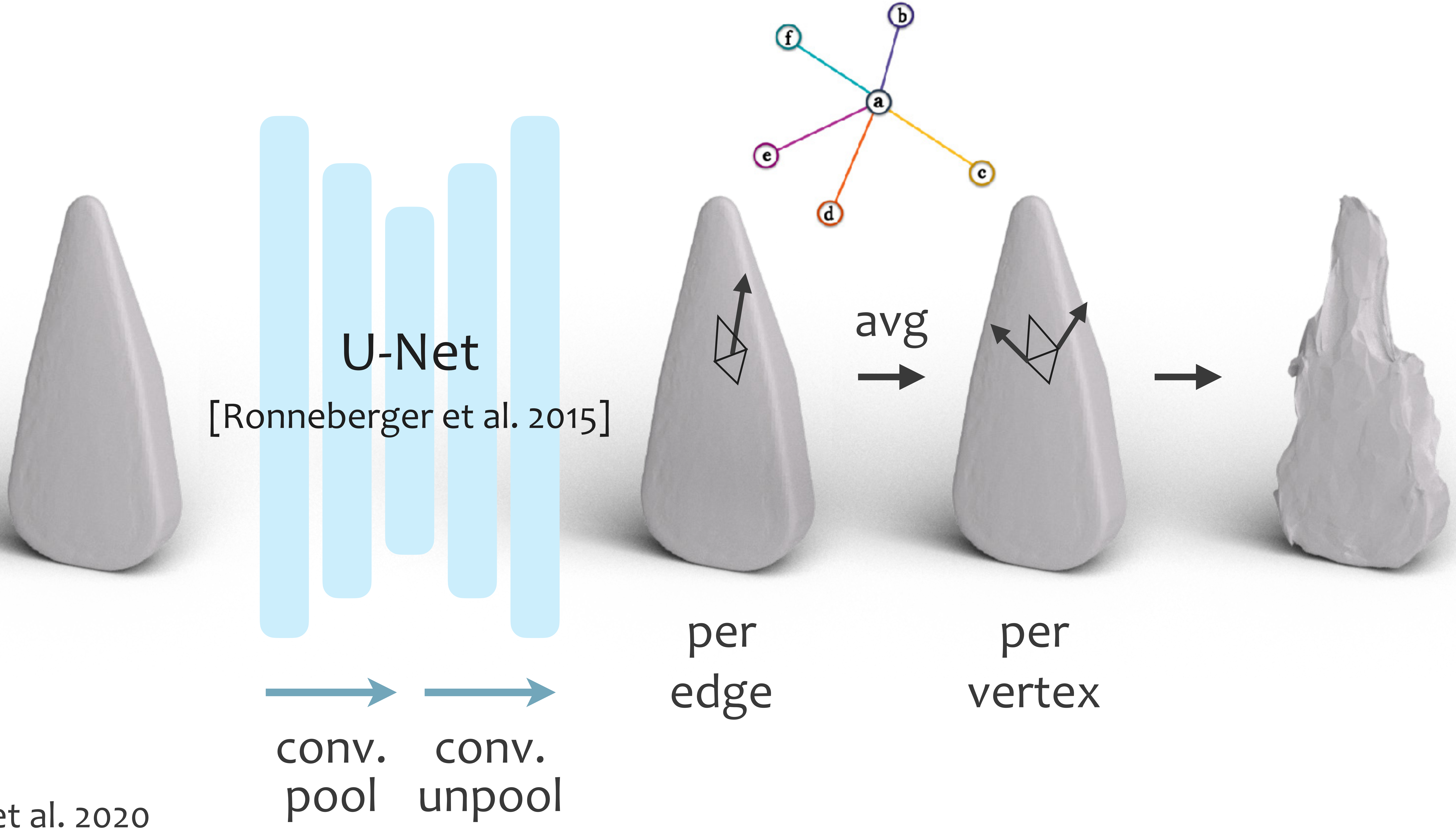
Hanocka et al. 2020

Point2Mesh Overview



Hanocka et al. 2020

Point2Mesh Overview



U-Net

[Ronneberger et al. 2015]

conv.
pool

conv.
unpool

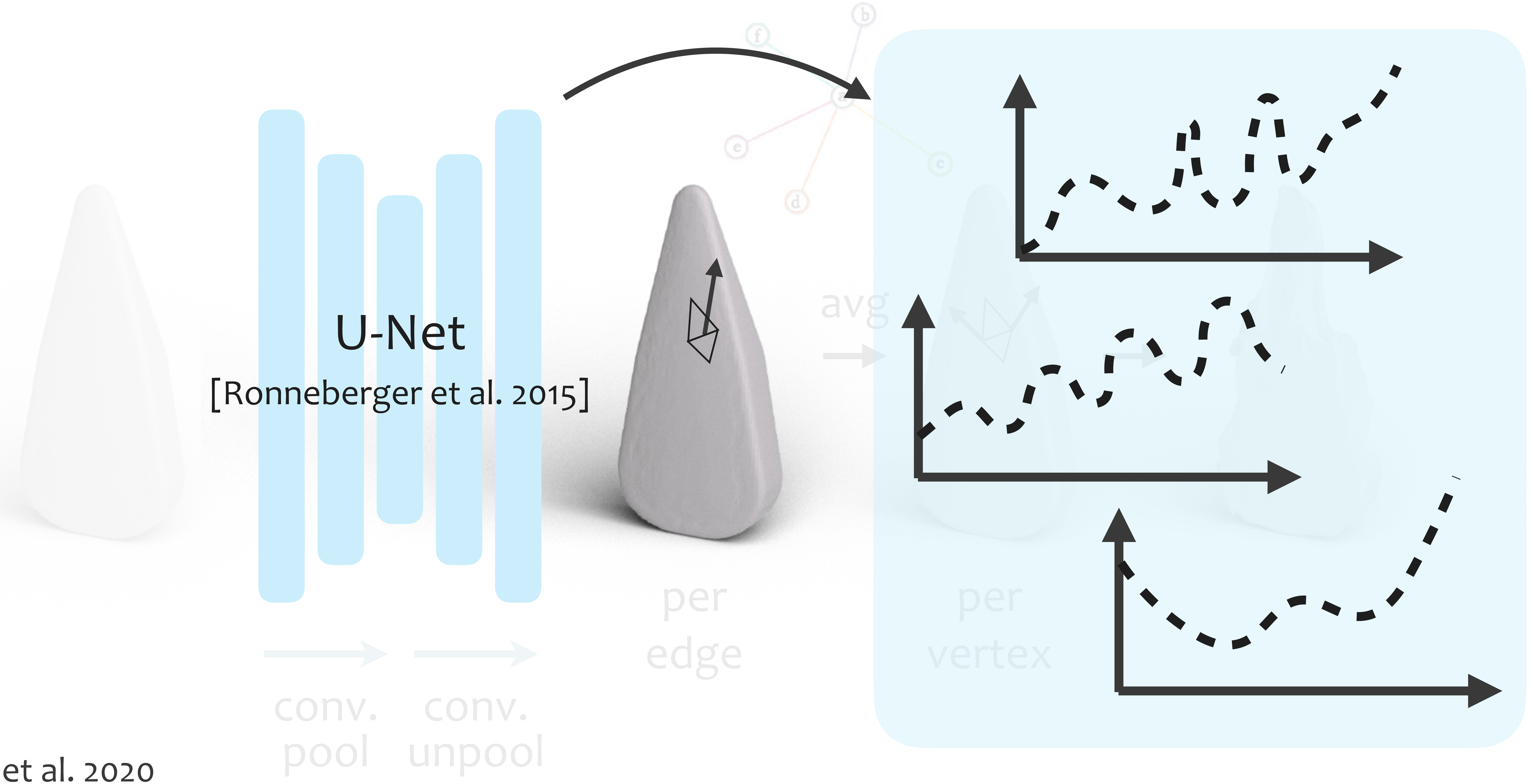
per
edge

avg

per
vertex

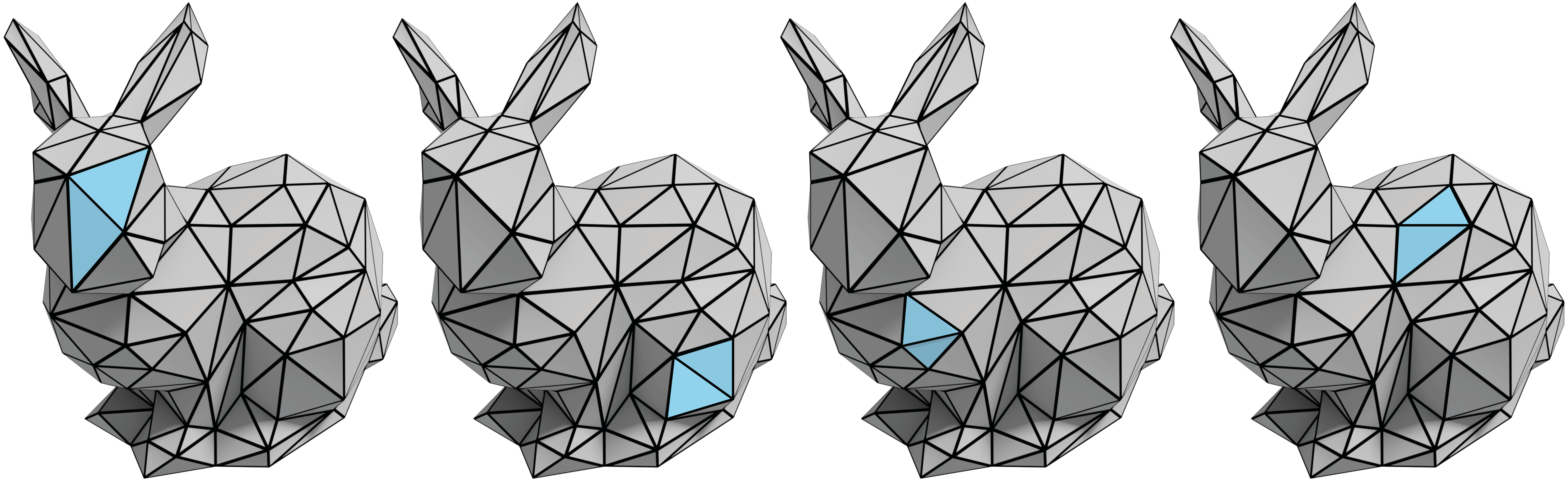
Hanocka et al. 2020

Point2Mesh Overview

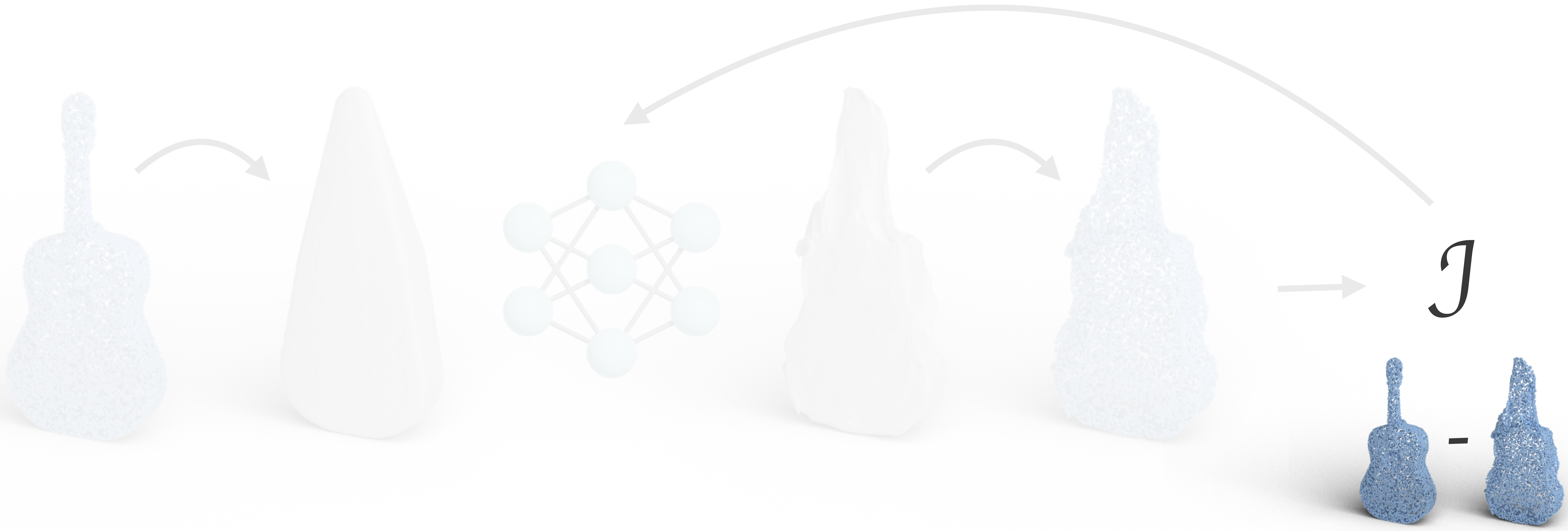


Hanocka et al. 2020

CNN filters are shared

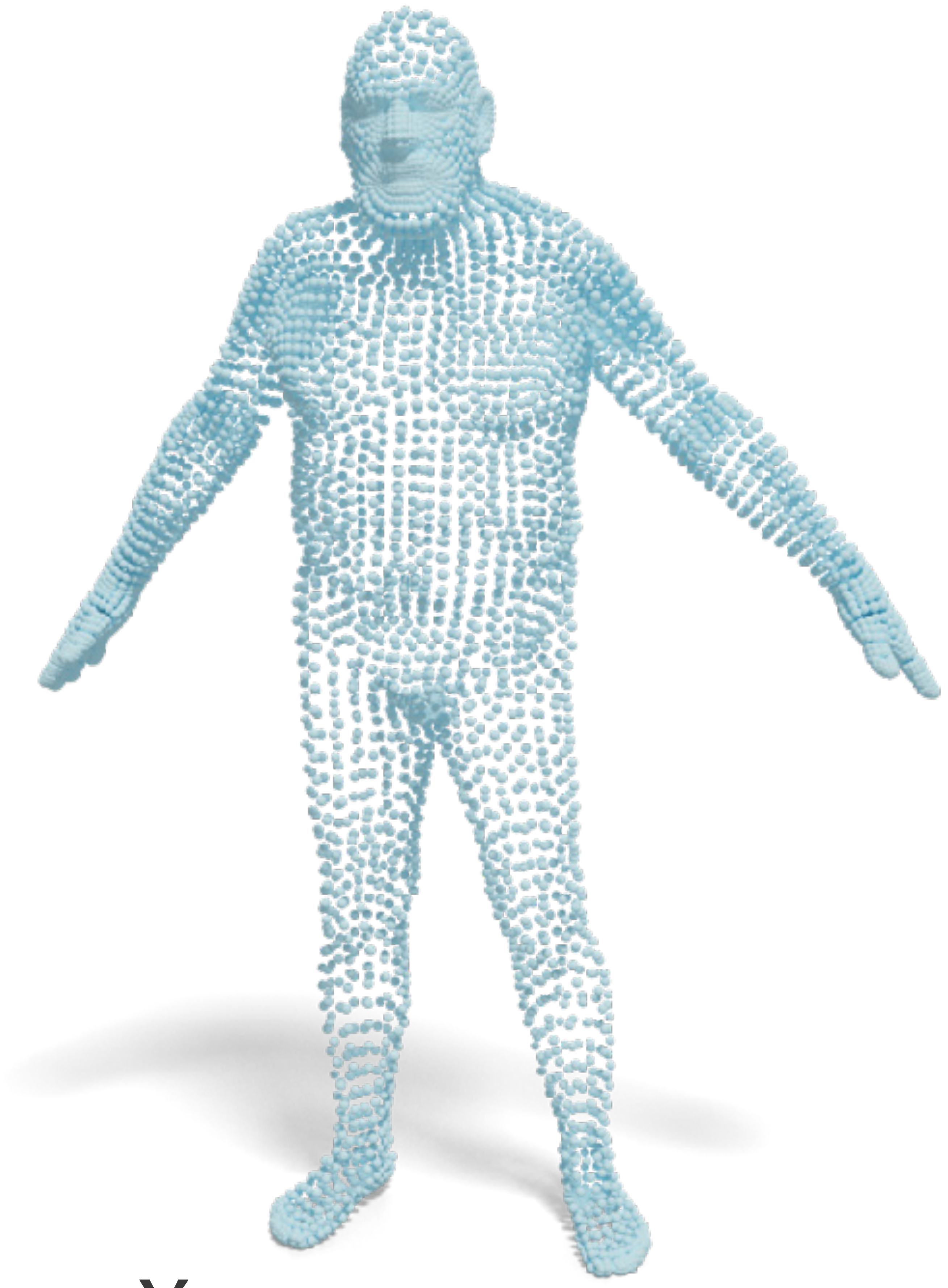


Point2Mesh Overview

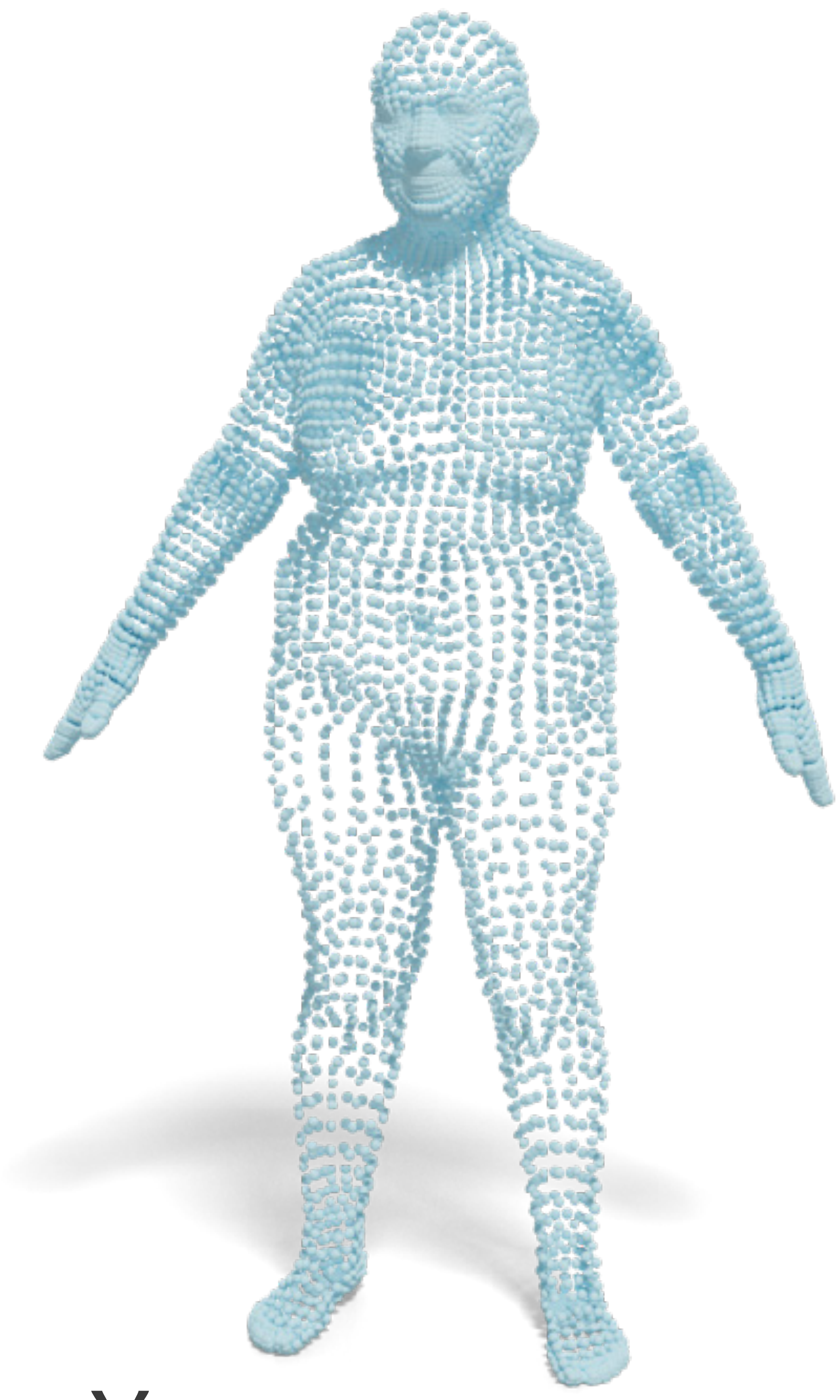


Hanocka et al. 2020

Loss Function: Chamfer Distance



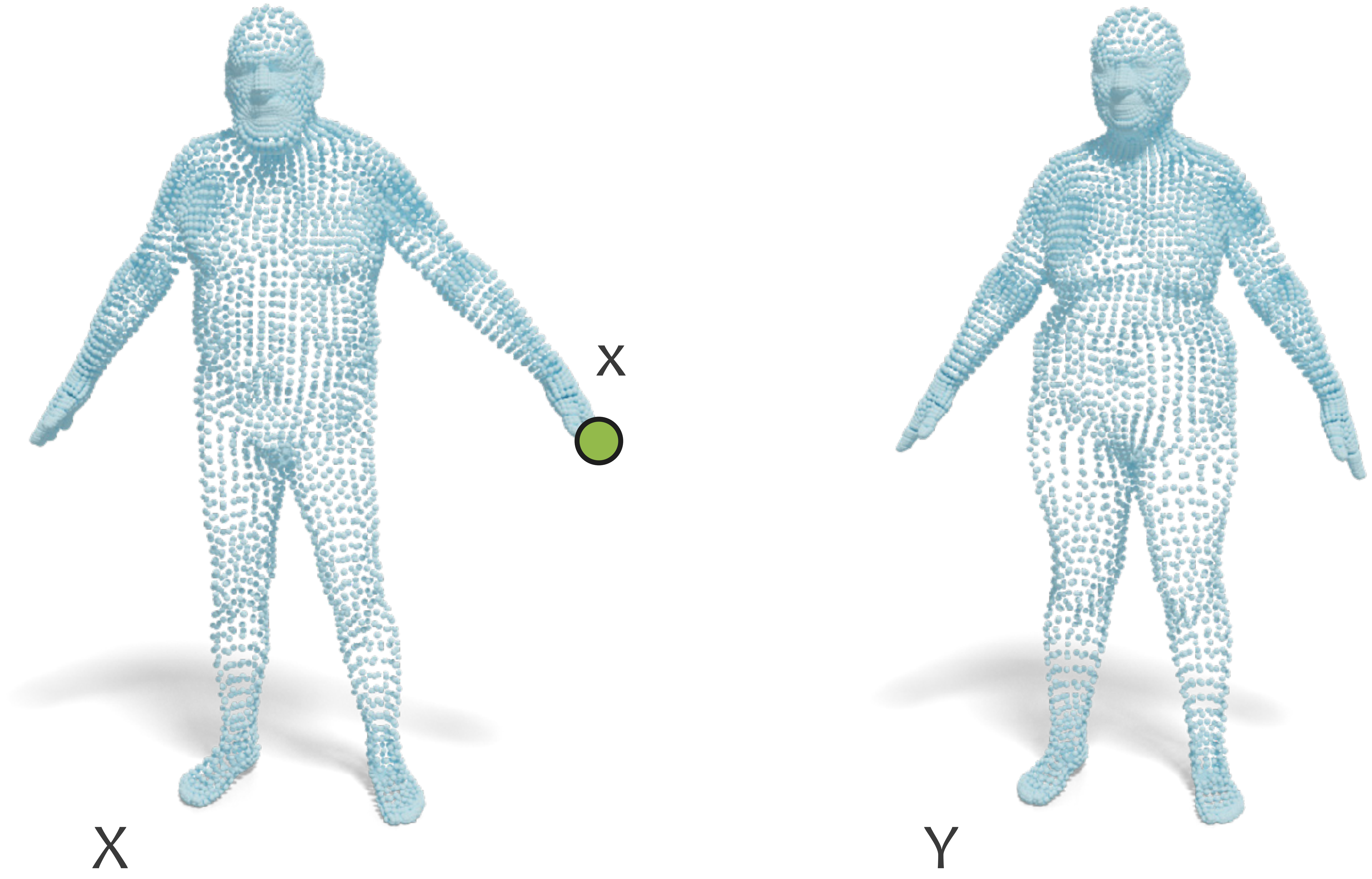
X



Y

Barrow et. al. 1977, Fan et. al. 2017

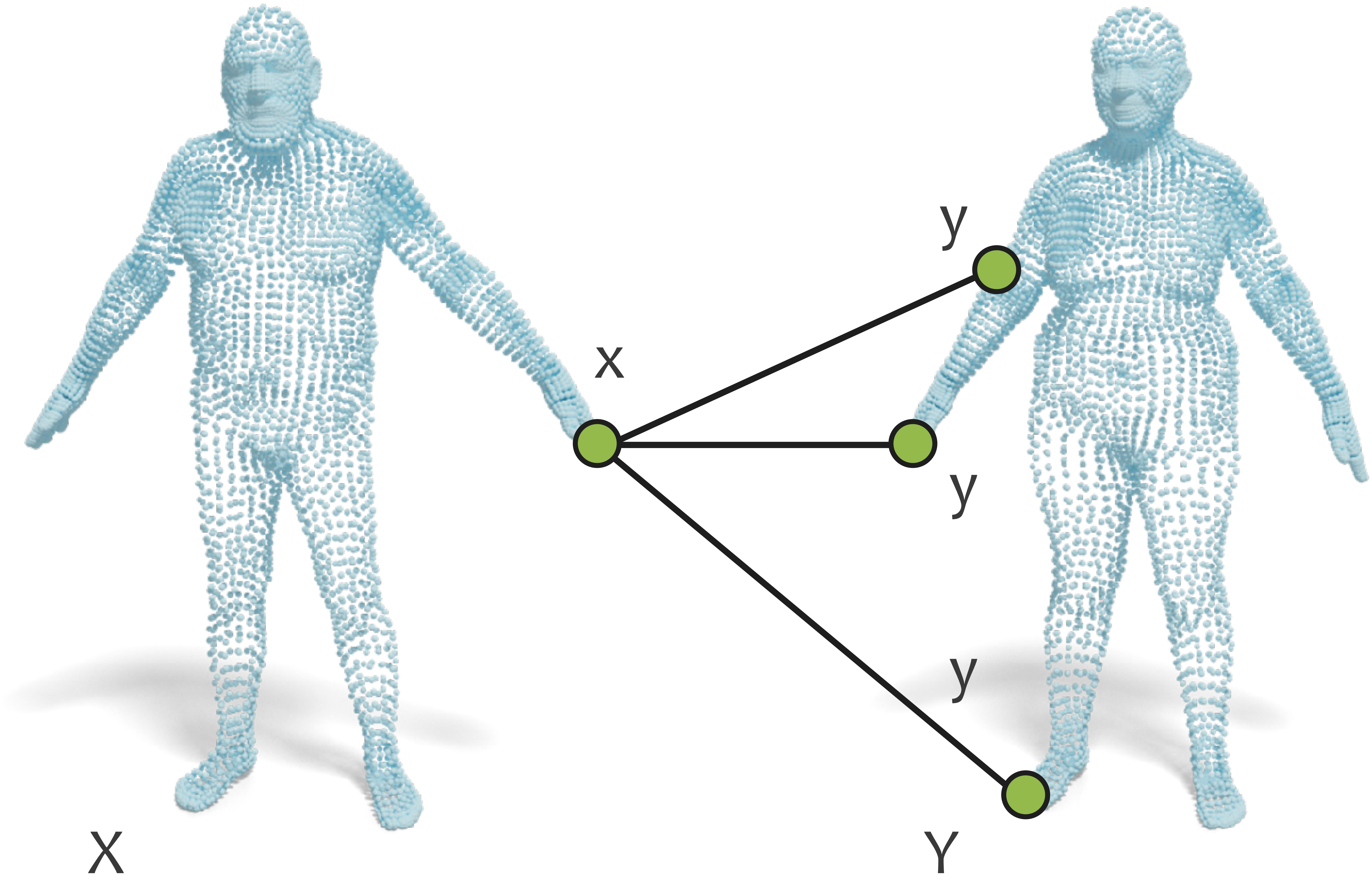
Loss Function: Chamfer Distance



$$\mathcal{J}(X, Y) = \qquad\qquad\qquad x$$

Barrow et. al. 1977, Fan et. al. 2017

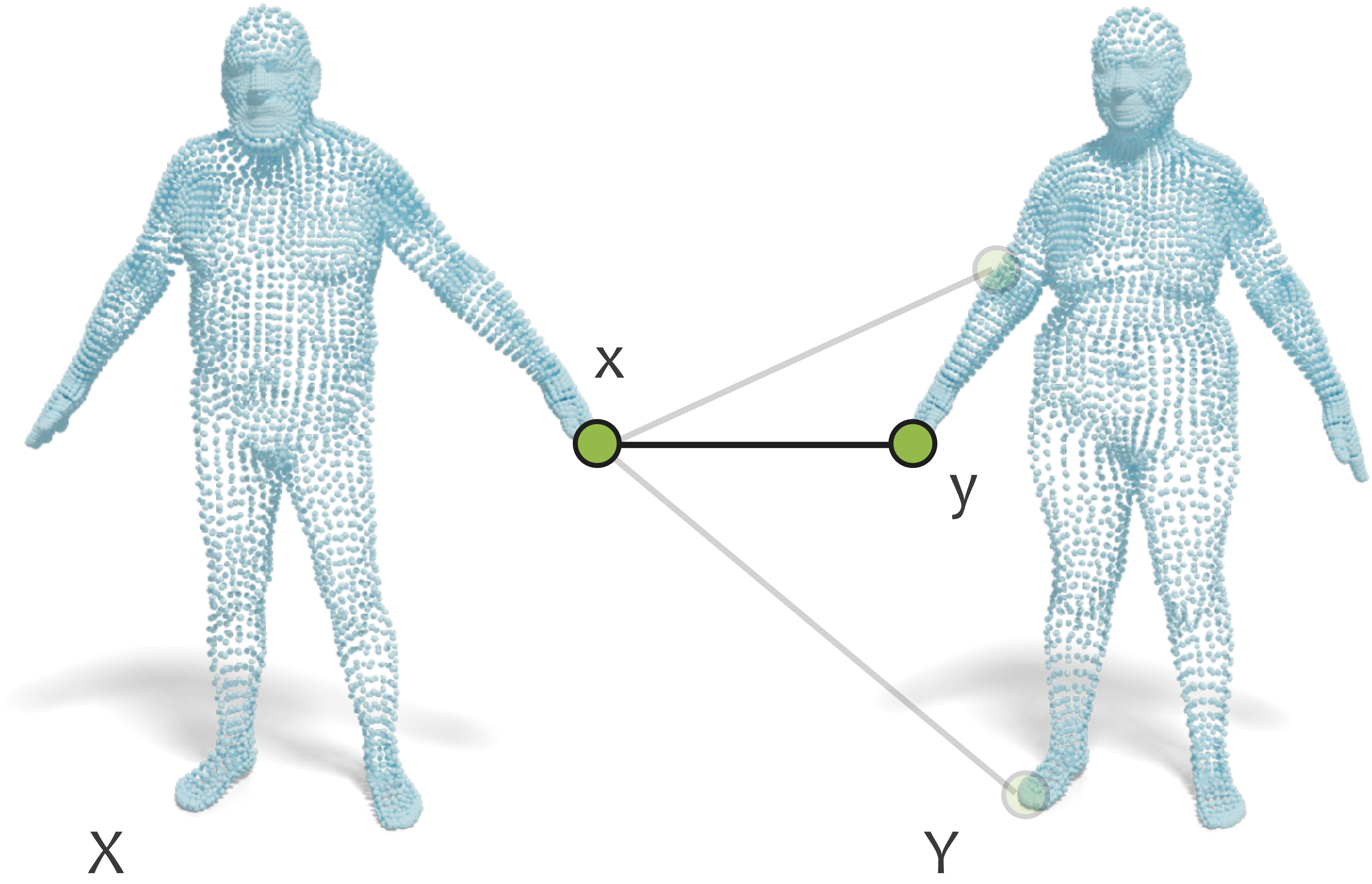
Loss Function: Chamfer Distance



$$\mathcal{J}(X, Y) = \qquad\qquad \|x - y\|^2$$

Barrow et. al. 1977, Fan et. al. 2017

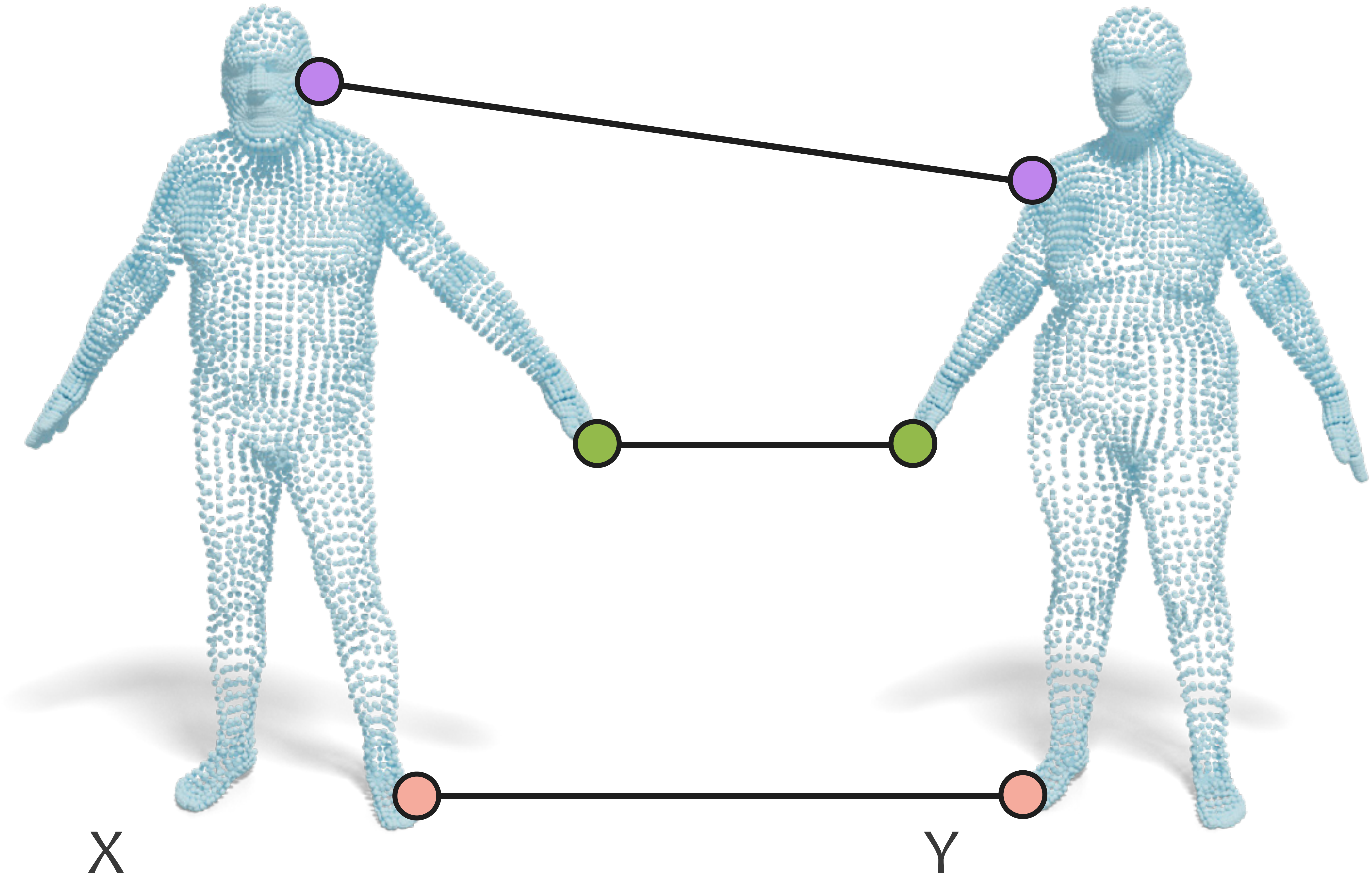
Loss Function: Chamfer Distance



$$\mathcal{J}(X, Y) = \min_{y \in Y} \|x - y\|^2$$

Barrow et. al. 1977, Fan et. al. 2017

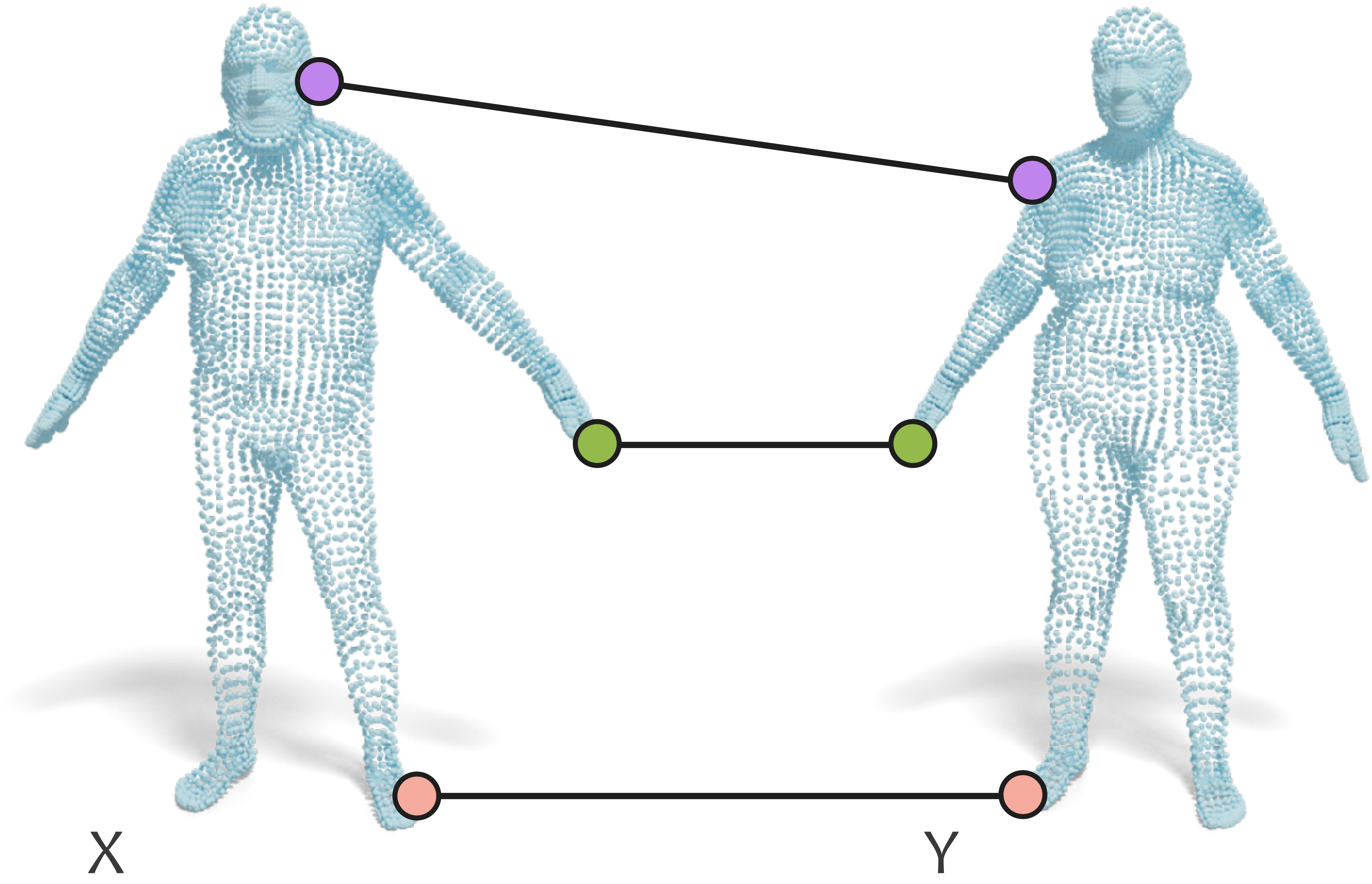
Loss Function: Chamfer Distance



$$\mathcal{J}(X, Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|^2$$

Barrow et. al. 1977, Fan et. al. 2017

Loss Function: Chamfer Distance

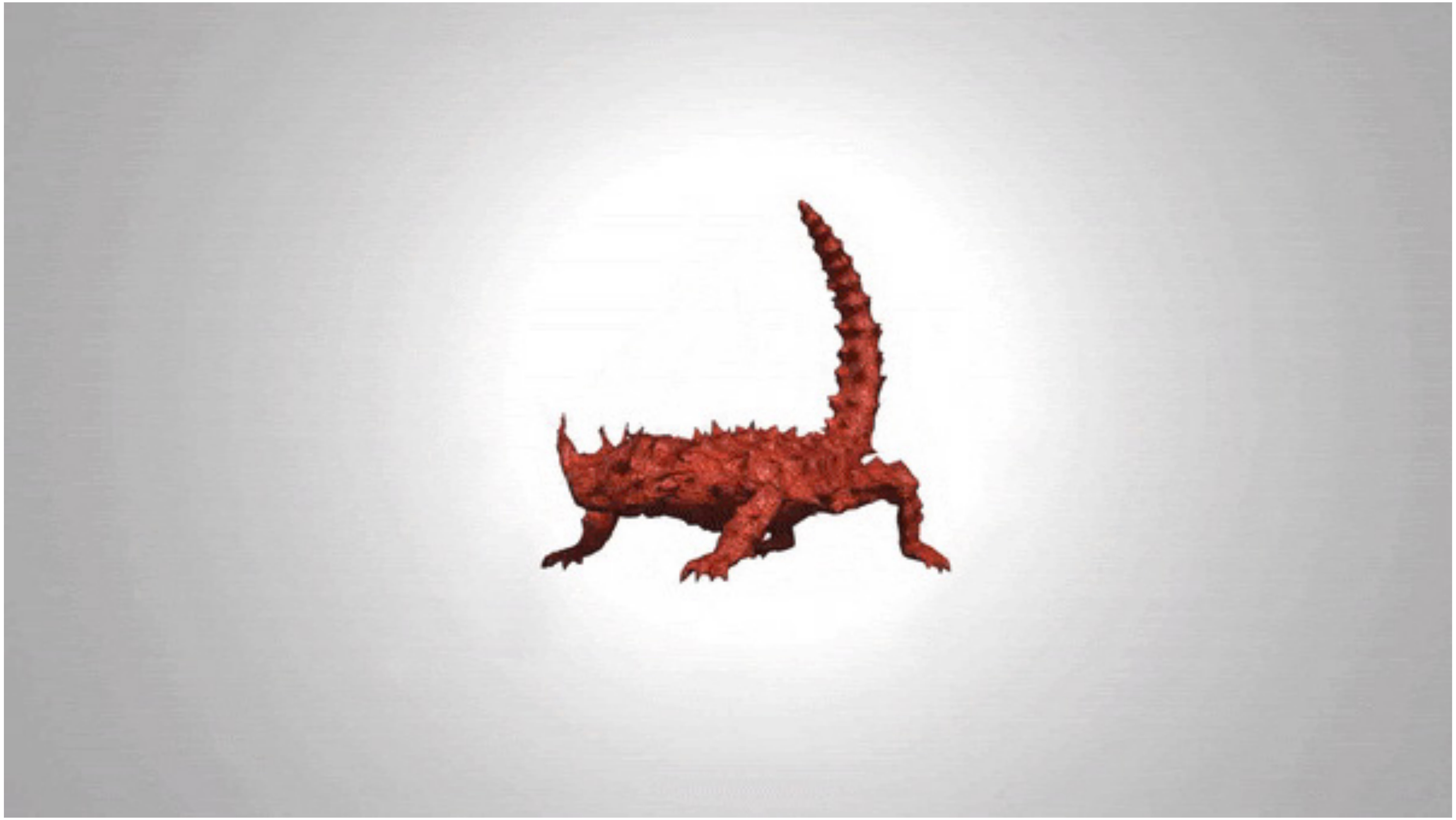


$$\mathcal{J}(X, Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|^2 + \sum_{y \in Y} \min_{x \in X} \|y - x\|^2$$

bidirectional Chamfer

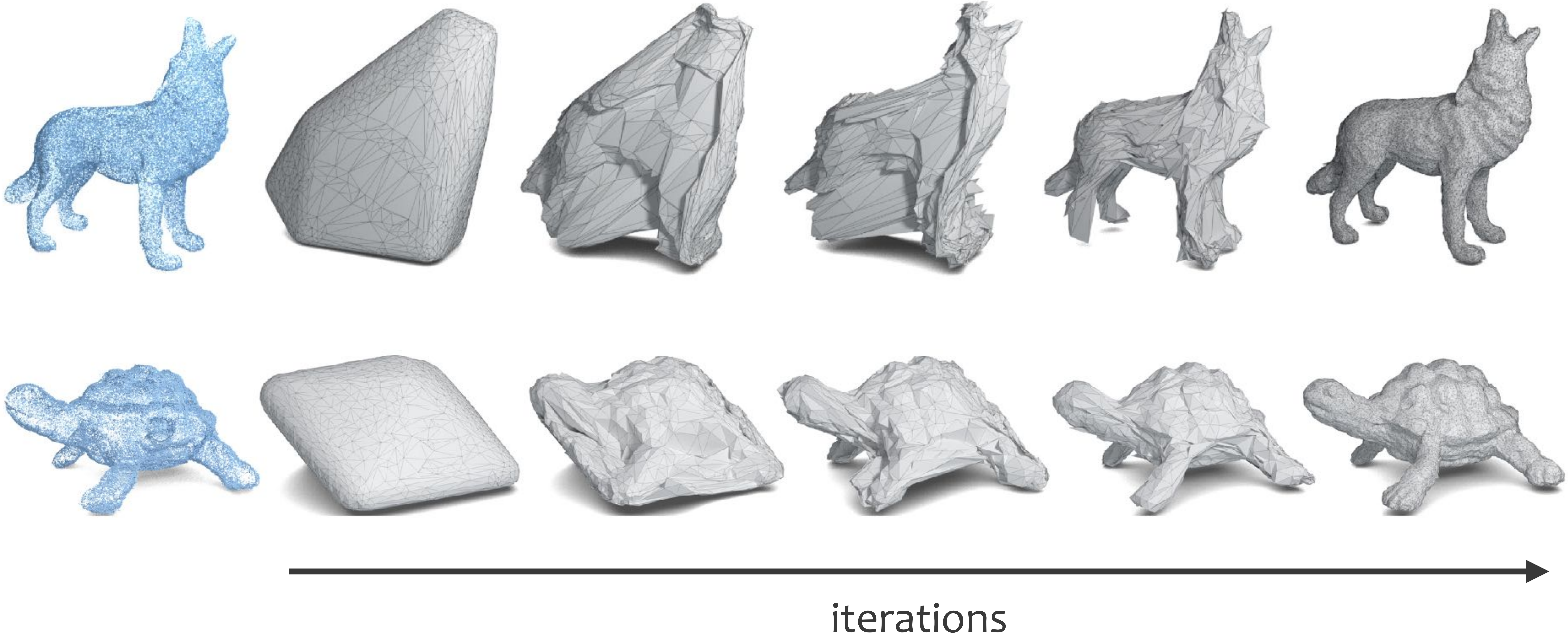
Barrow et. al. 1977, Fan et. al. 2017

Optimization



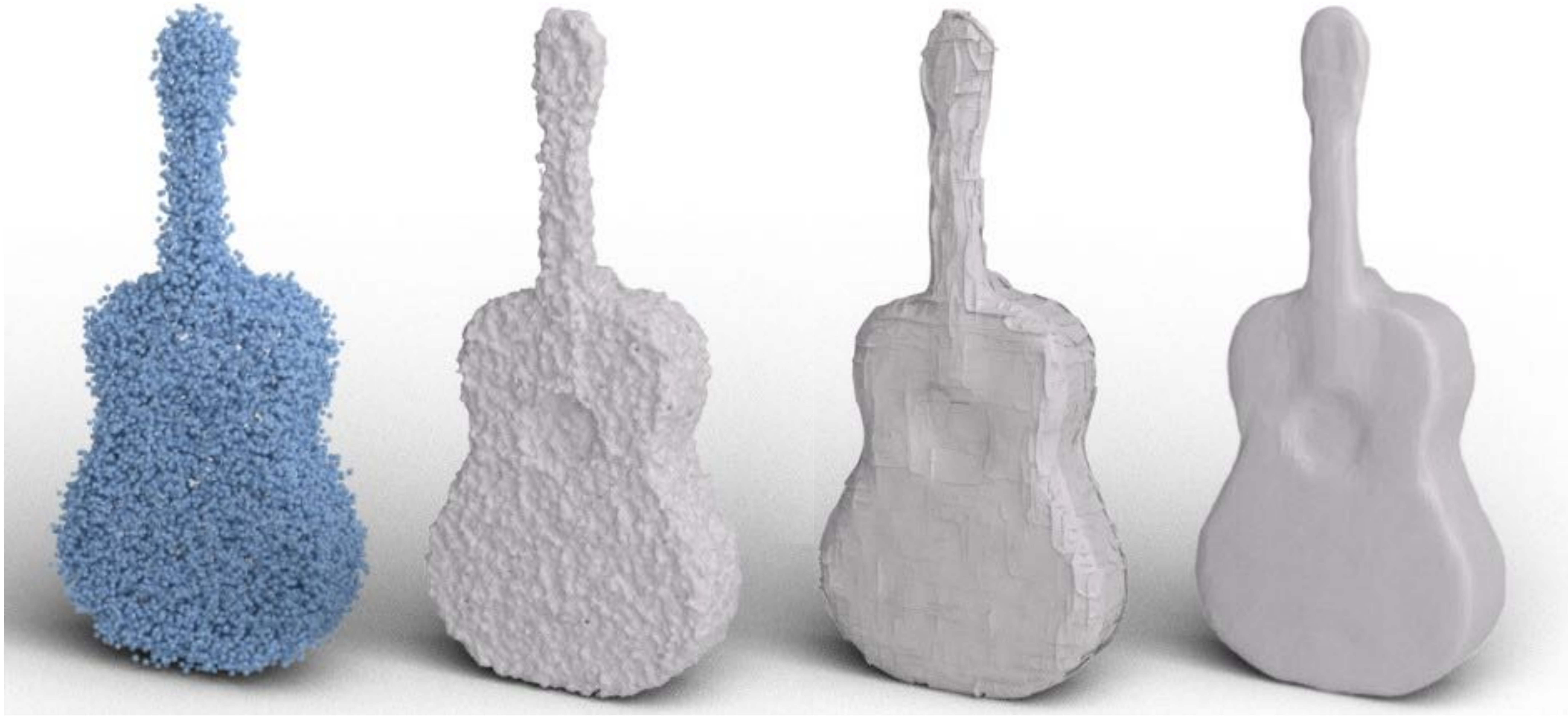
Hanocka et al. 2020

Results



Hanocka et al. 2020

Denosing



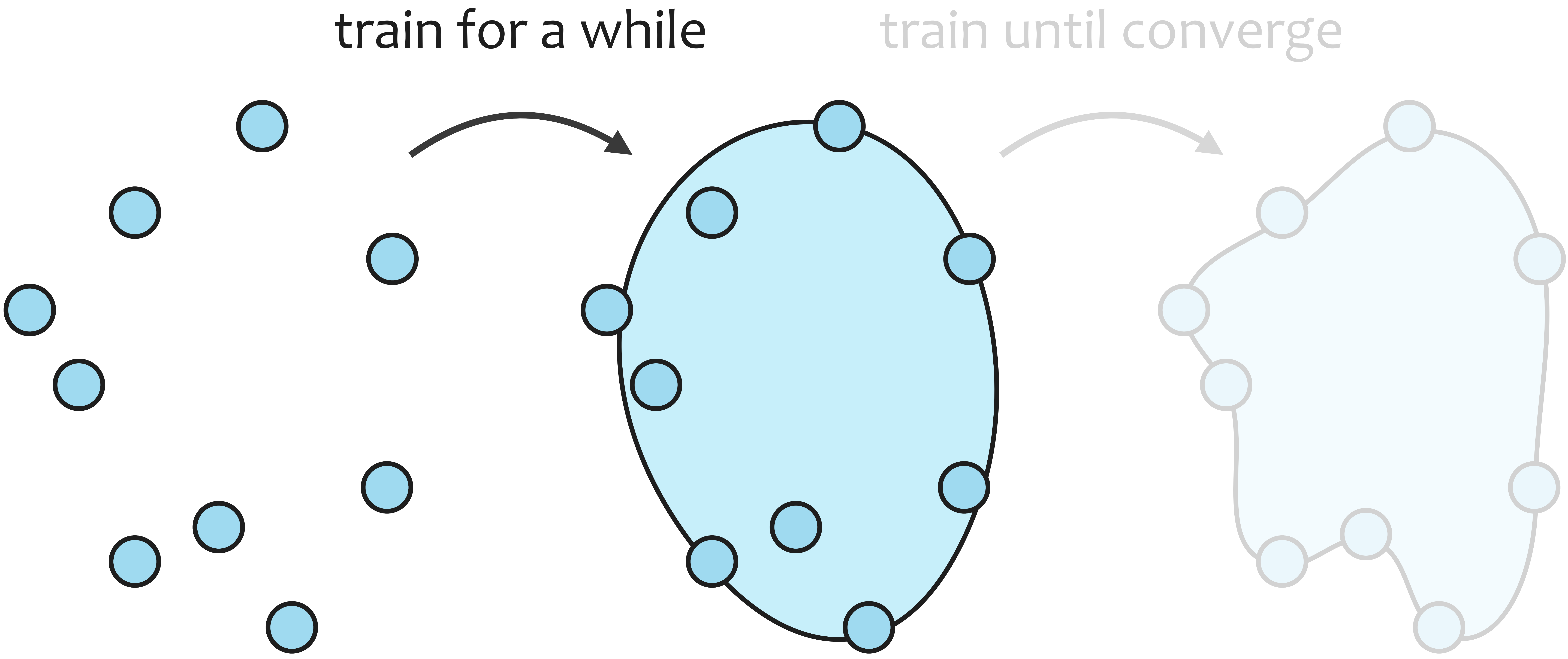
smoothness prior

deep geometric prior

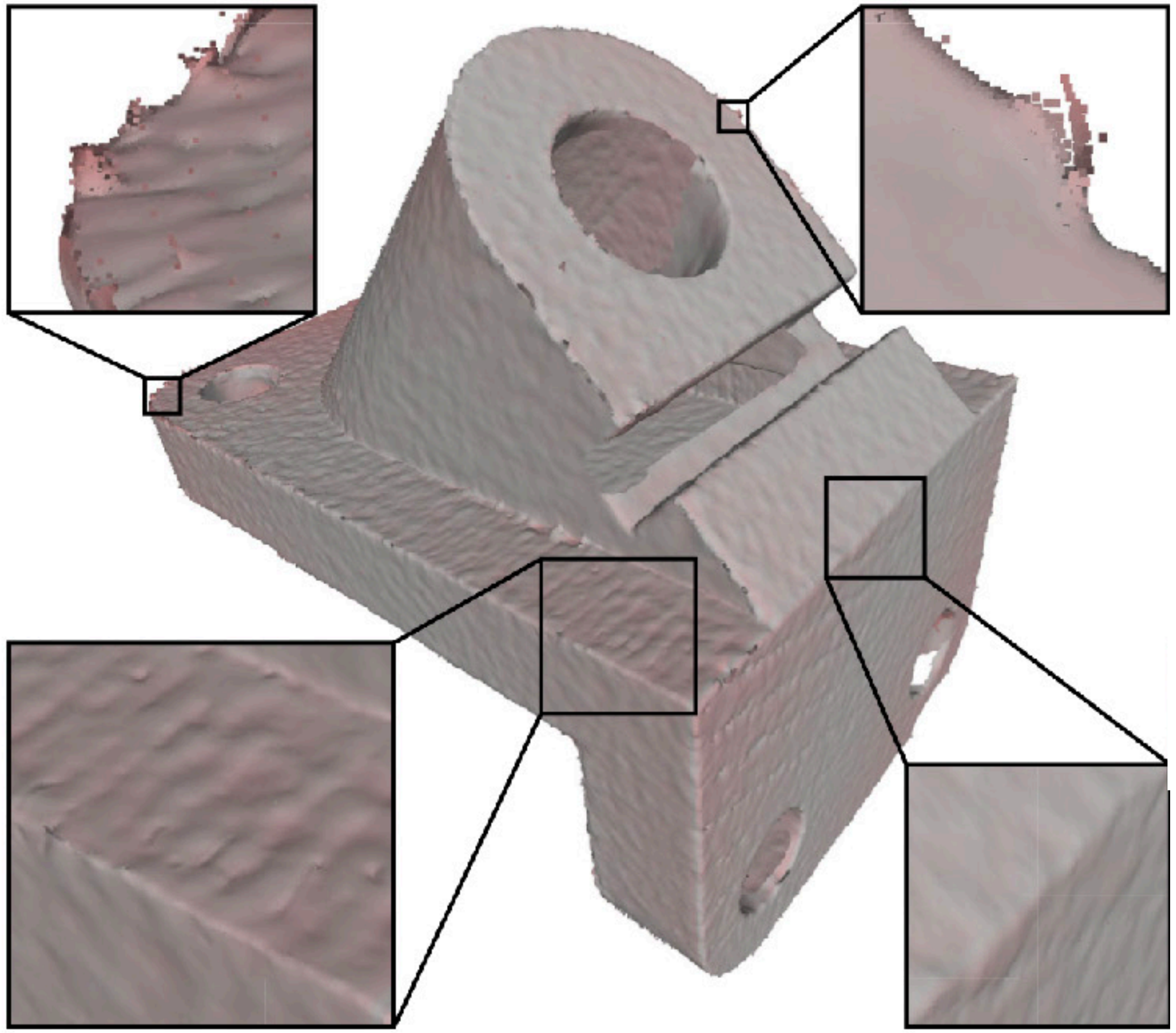
self-prior

Hanocka et al. 2020

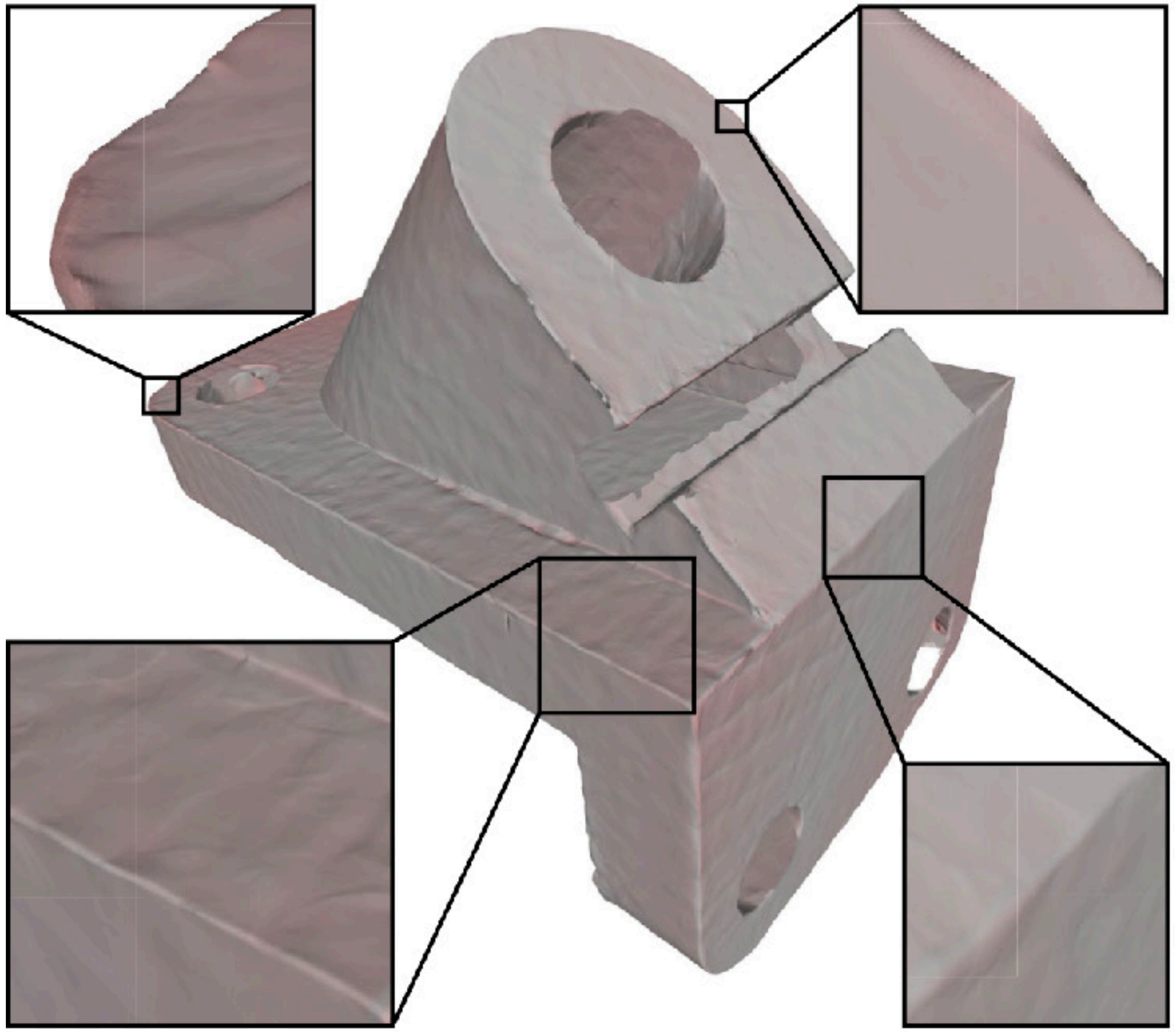
Early-stop as a regularization



Early-Stop Regularization



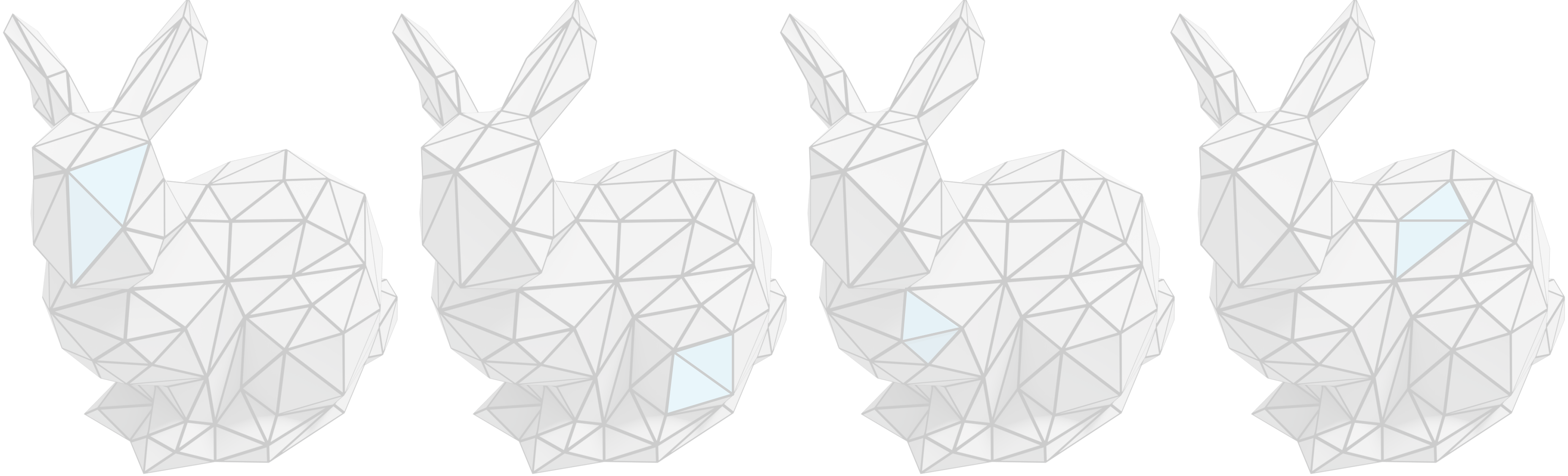
Classic



deep geometric prior

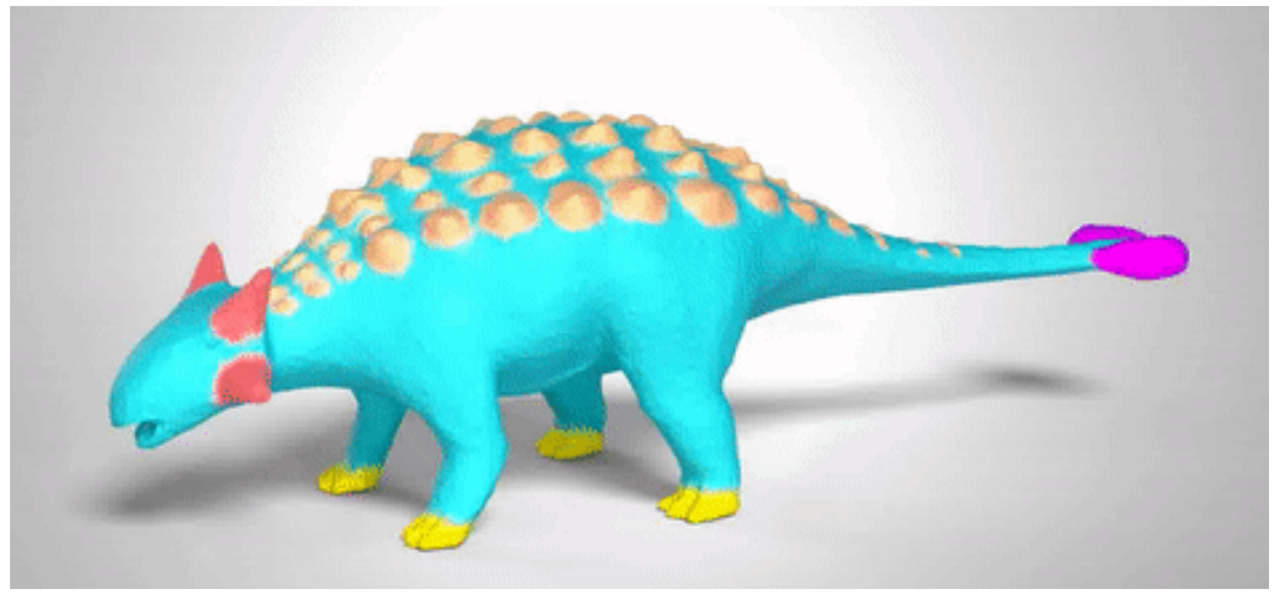
Williams et al. 2019

Mesh Convolutional Neural Networks



Machine Learning & Geometry Processing

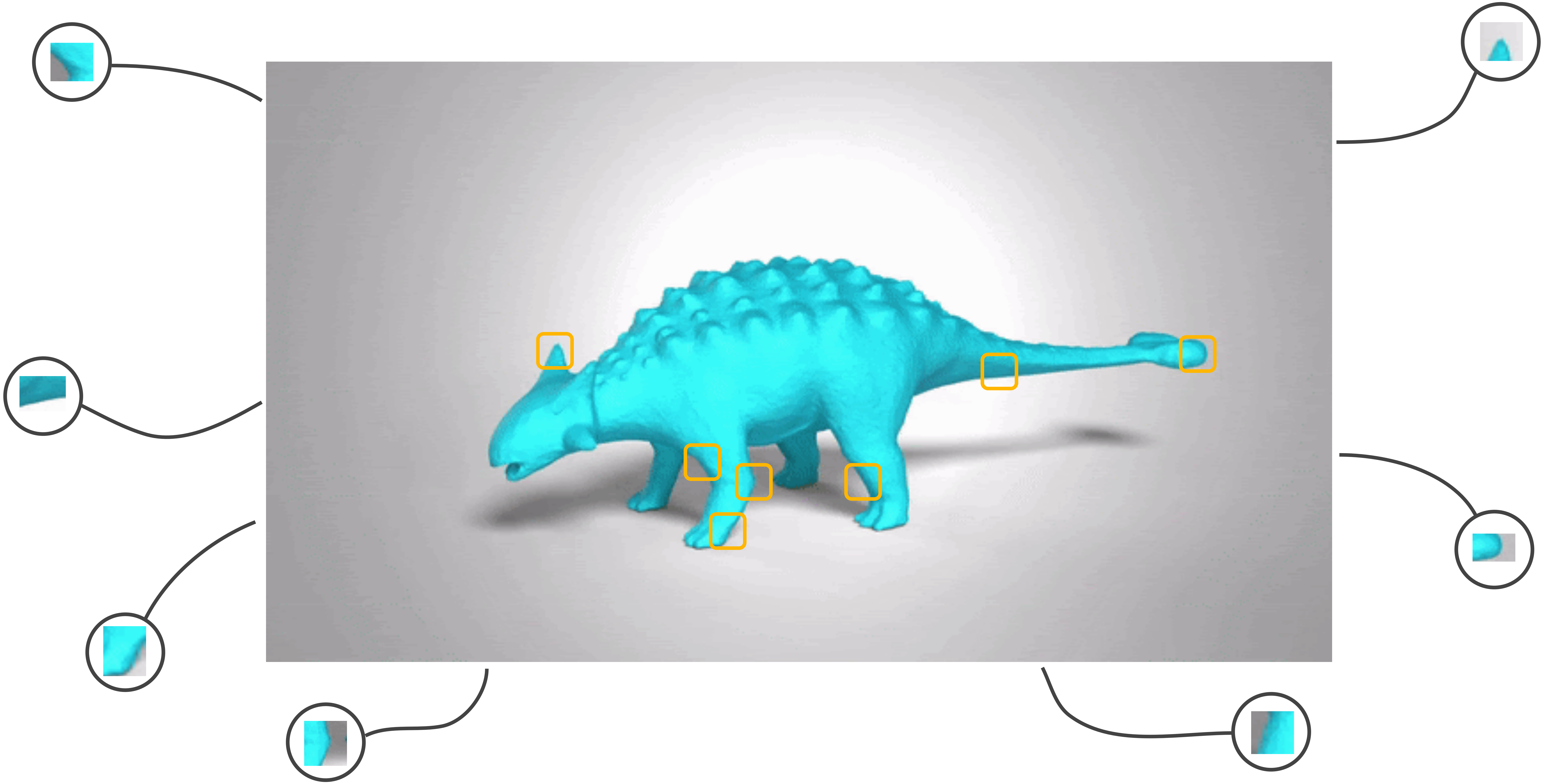
Learning from a Single Mesh



Learning from external data

source: NVIDIA kaolin

Learning from internal data



Example: Mesh Upsampling

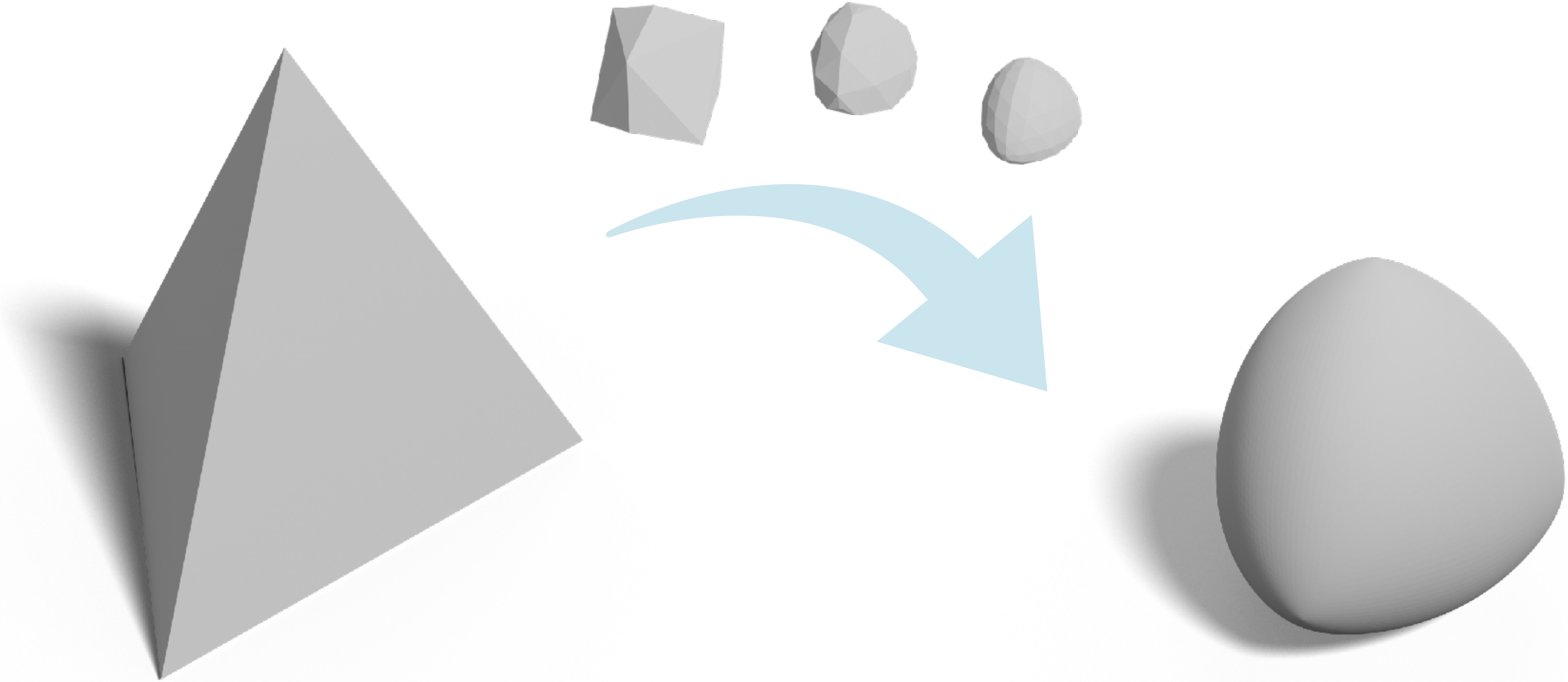


Image Upsampling

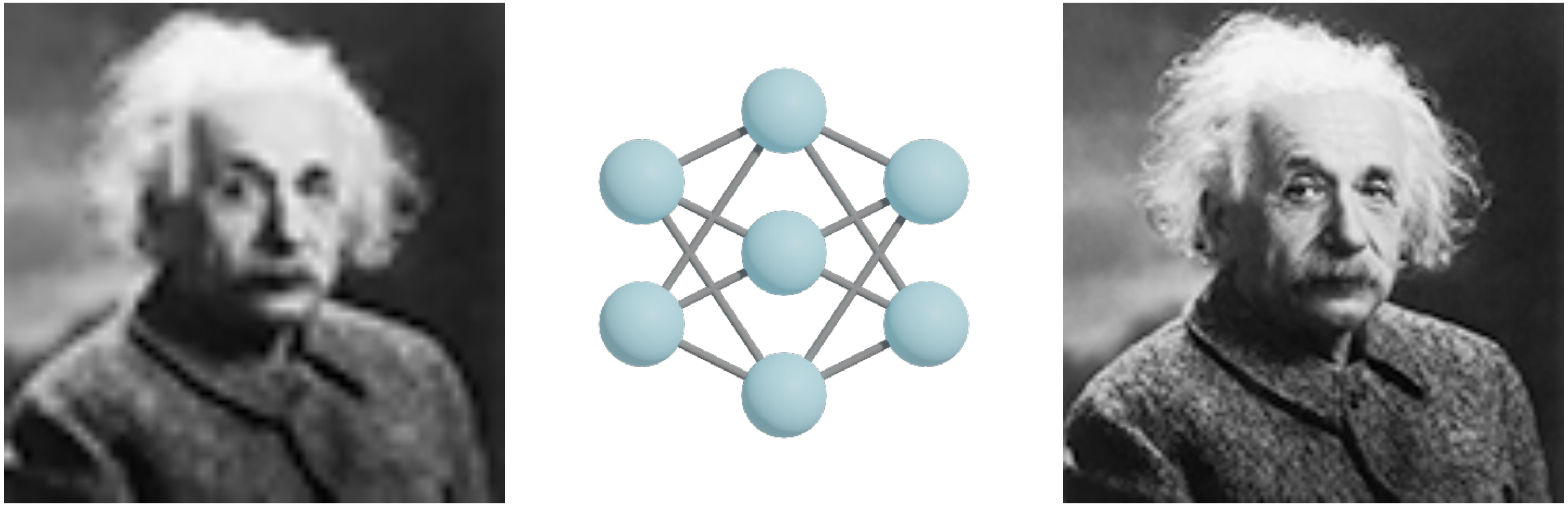
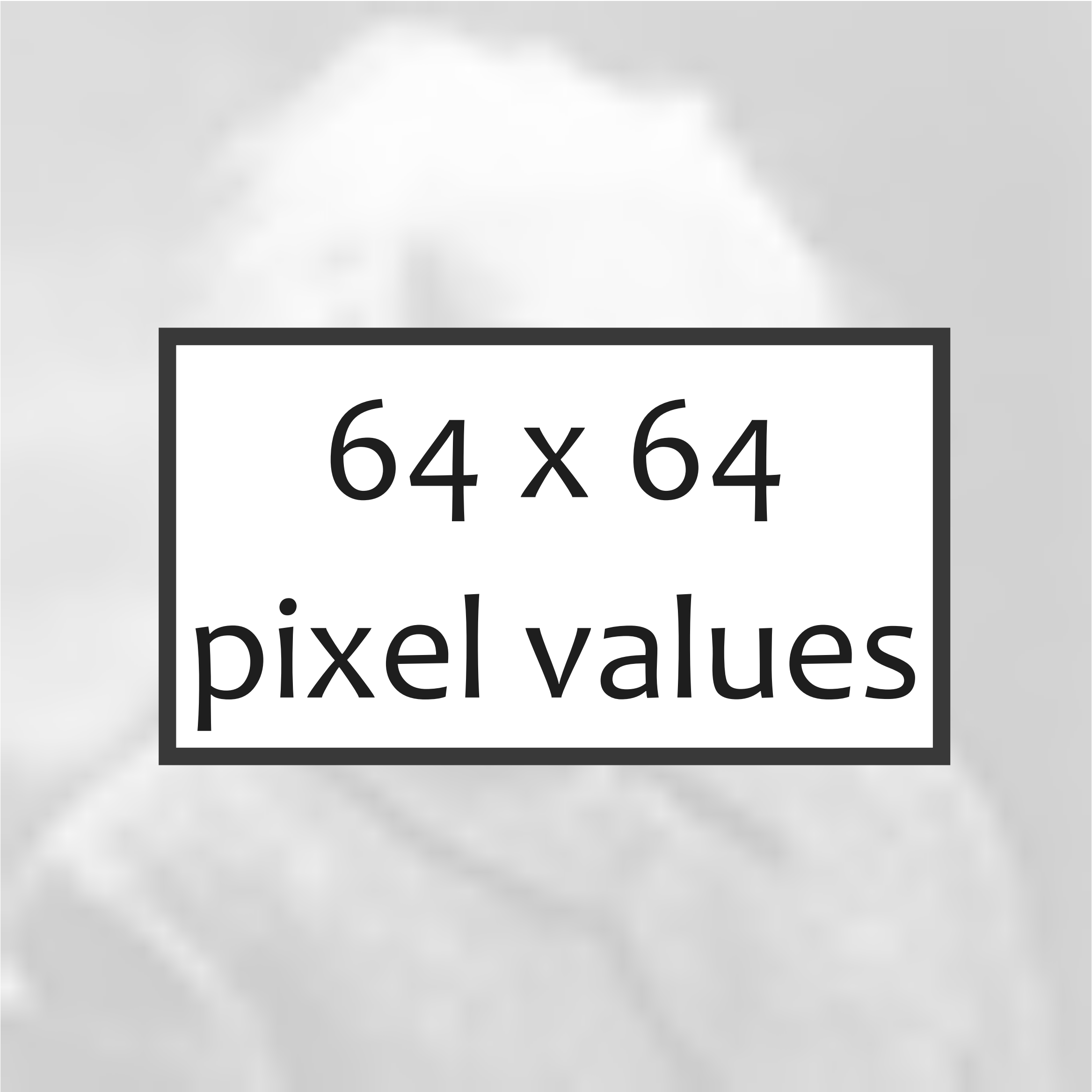
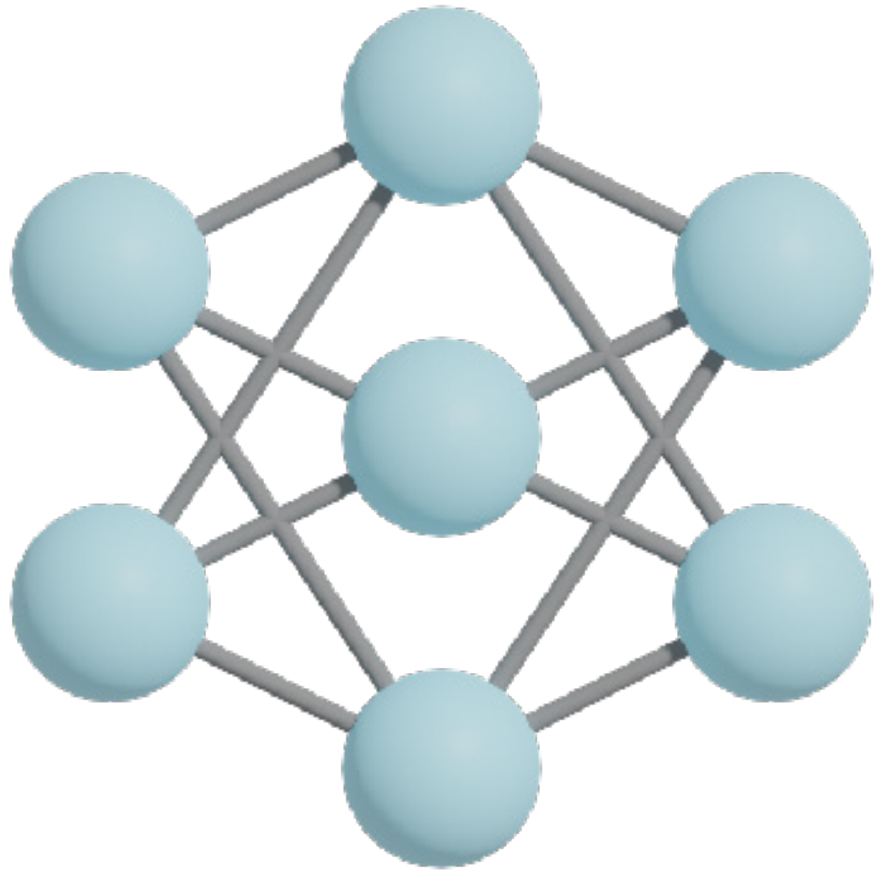
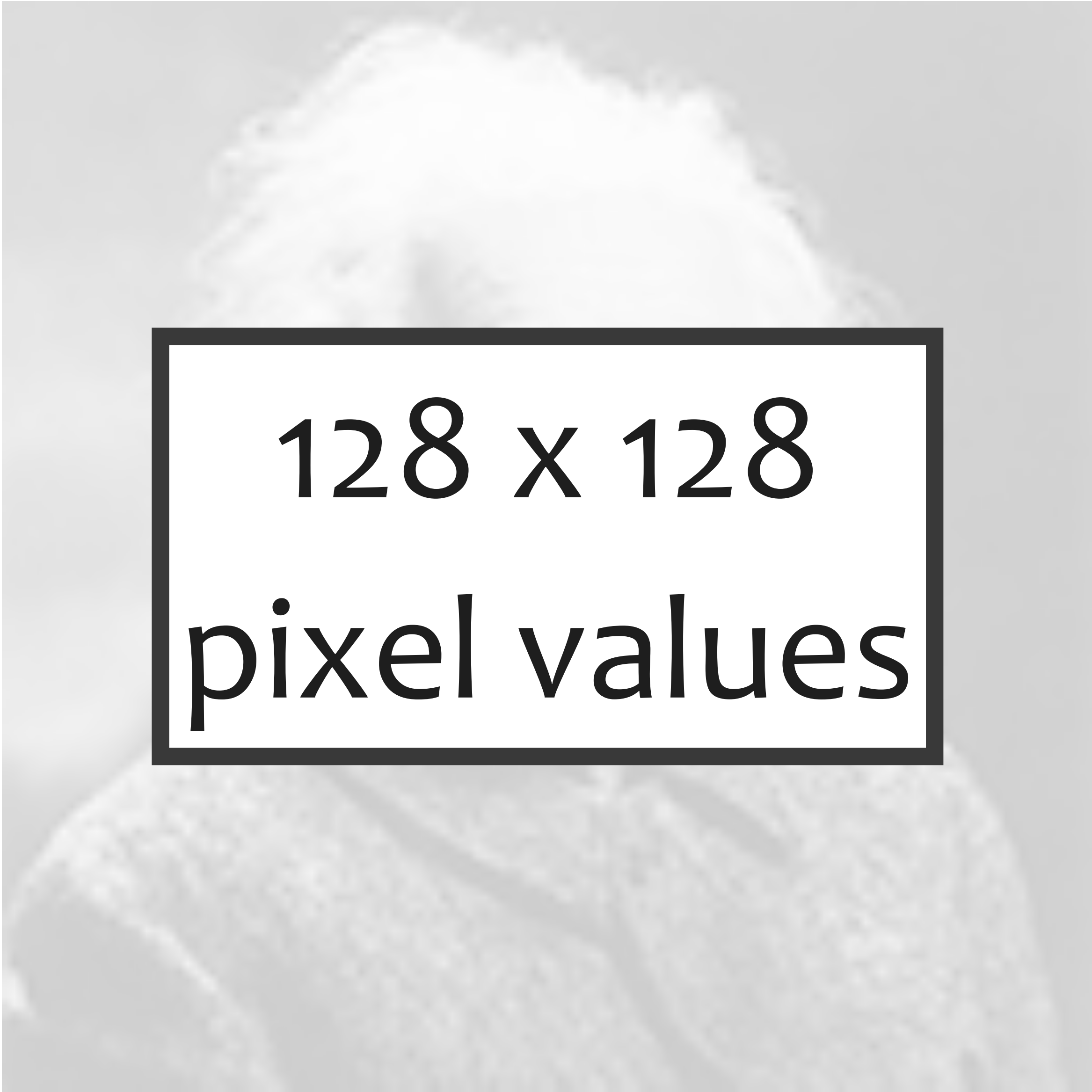


Image Upsampling



64 x 64
pixel values



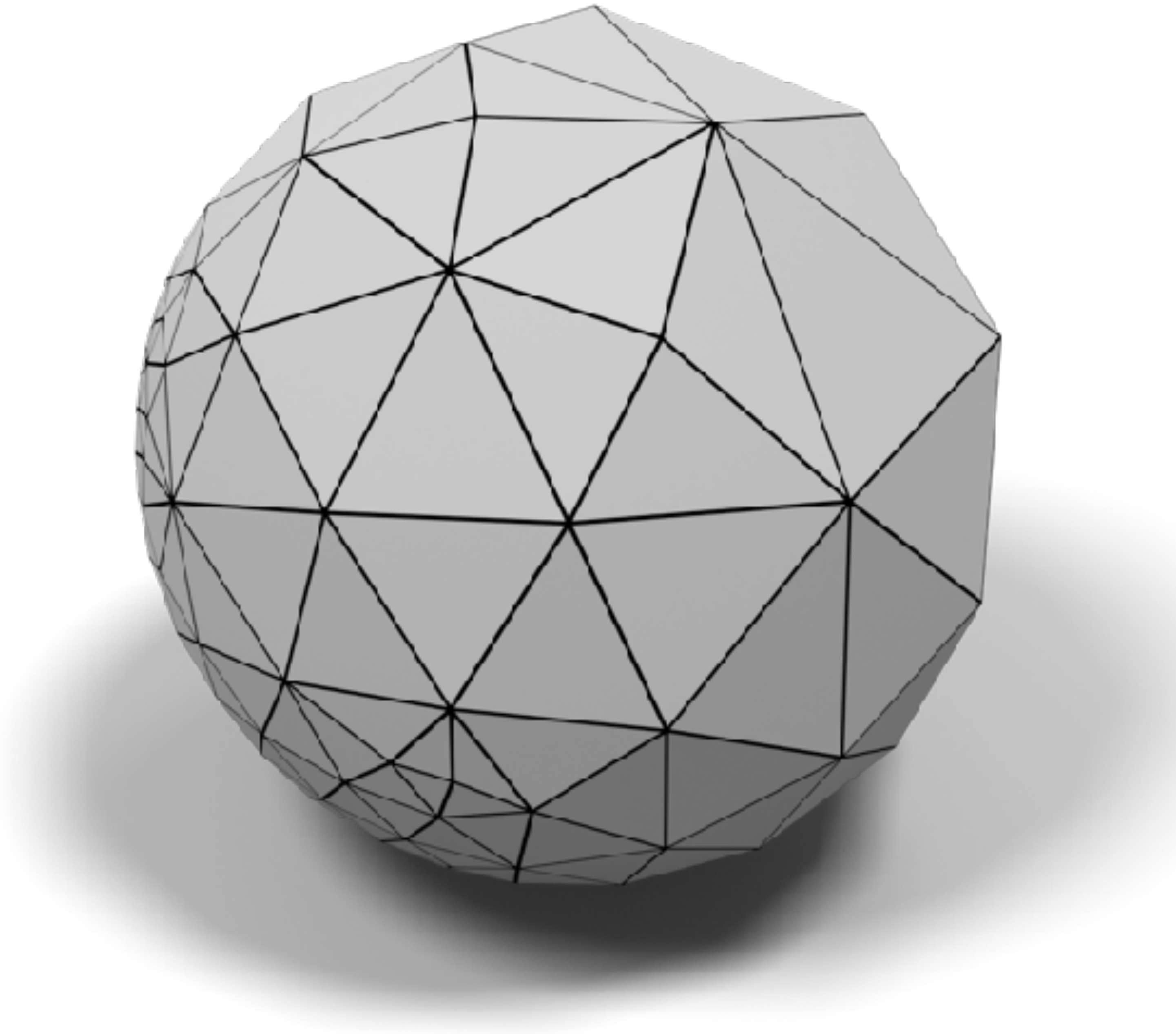


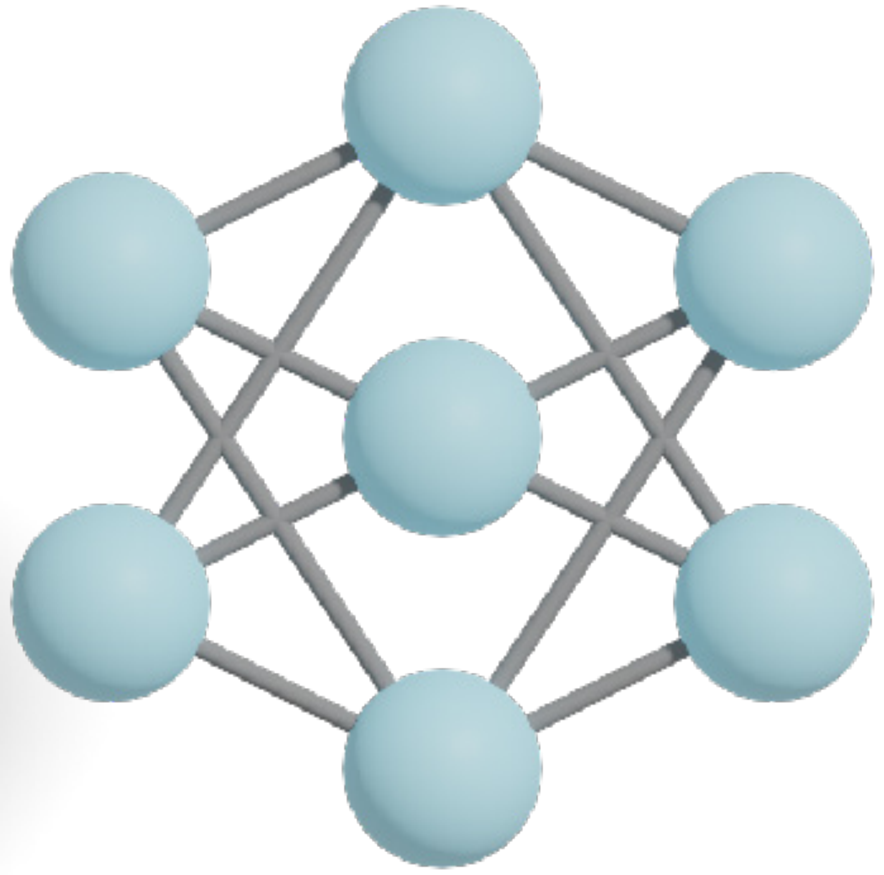
128 x 128
pixel values

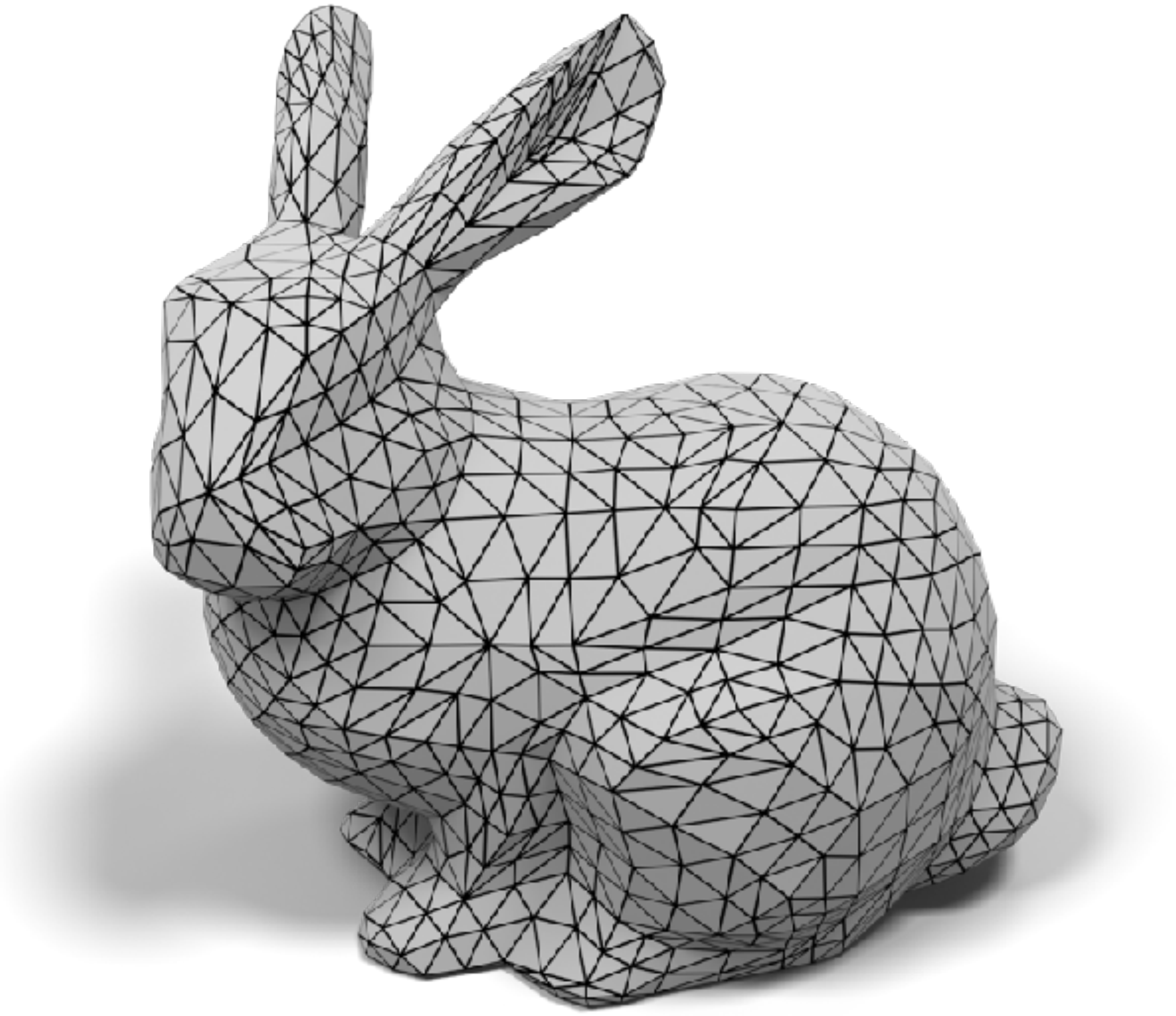
- ✓ assume image size 64
- ✓ assume grid structure
- ✓ assume top-left to bottom-right
- ✓ assume upright position

- ✓ assume image size 128
- ✓ assume grid structure
- ✓ assume top-left to bottom-right
- ✓ assume upright position

Draw inspiration from images



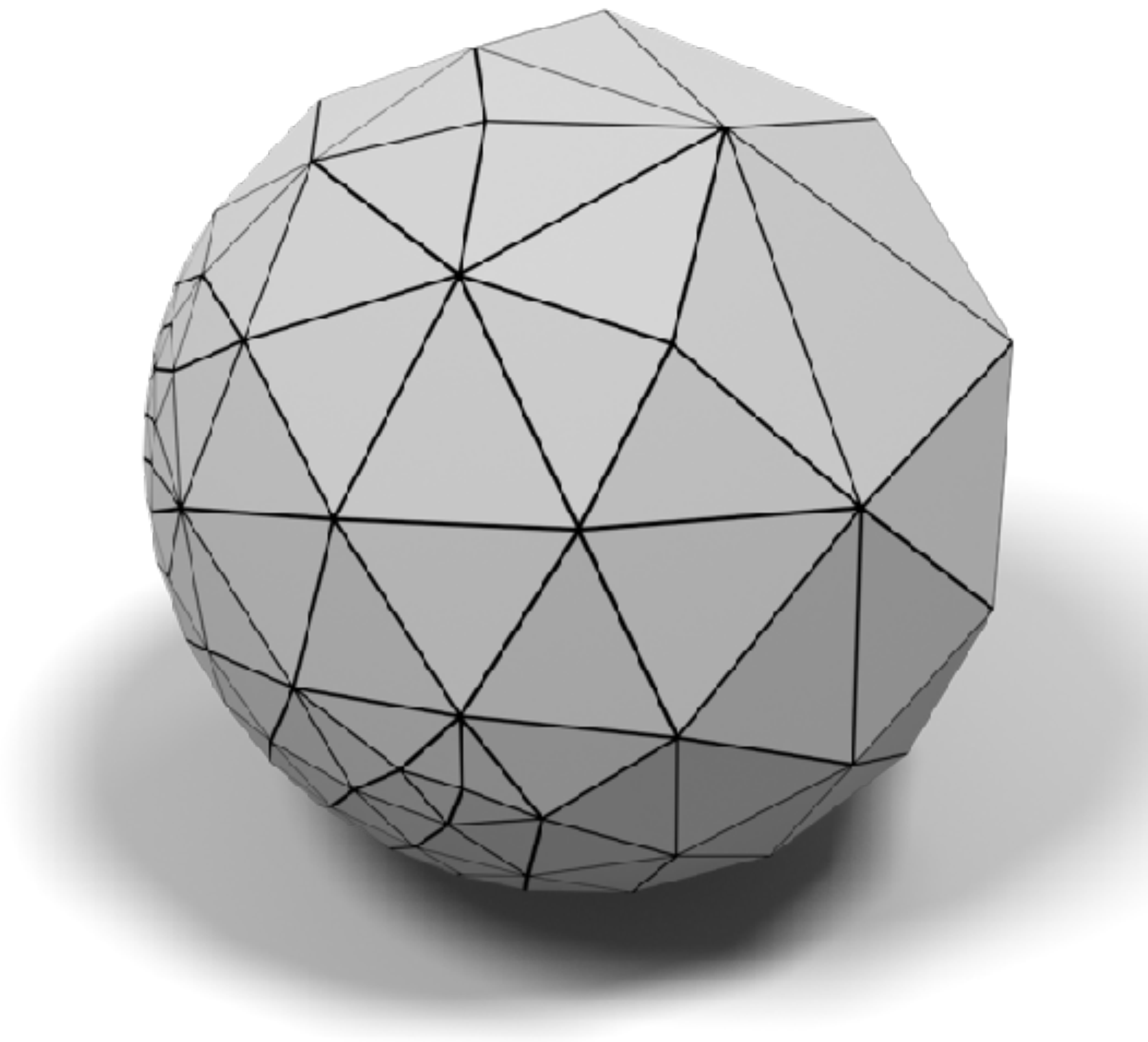


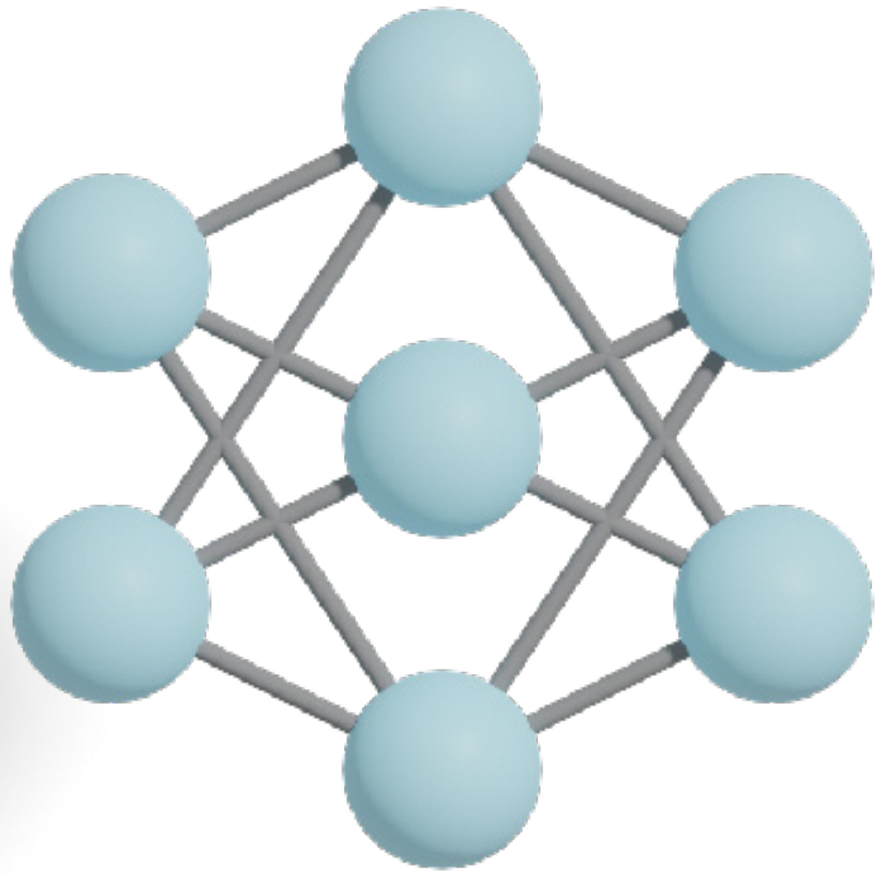


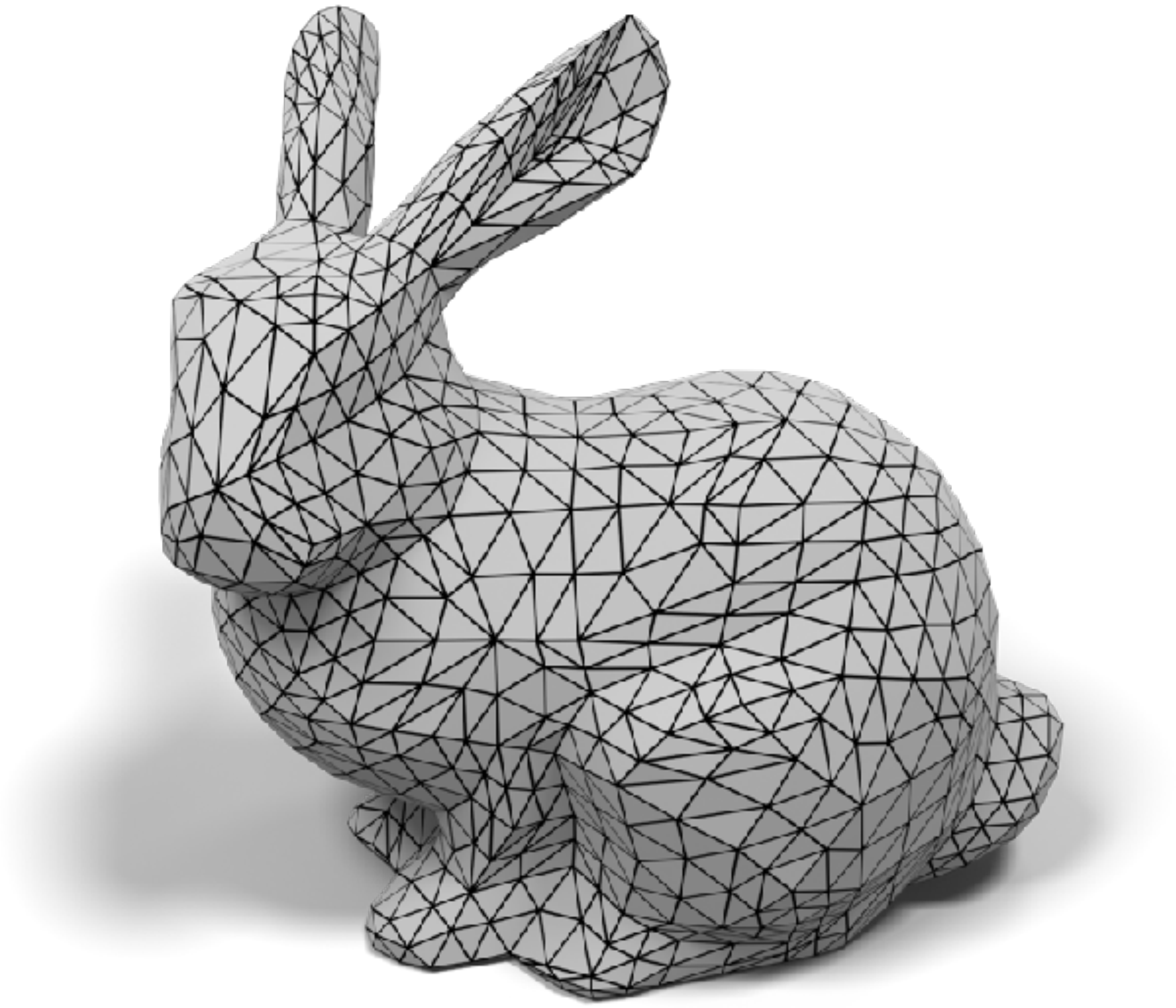
- assume image size 64
- assume grid structure
- assume top-left to bottom-right
- assume upright position

- assume image size 128
- assume grid structure
- assume top-left to bottom-right
- assume upright position

Draw inspiration from images



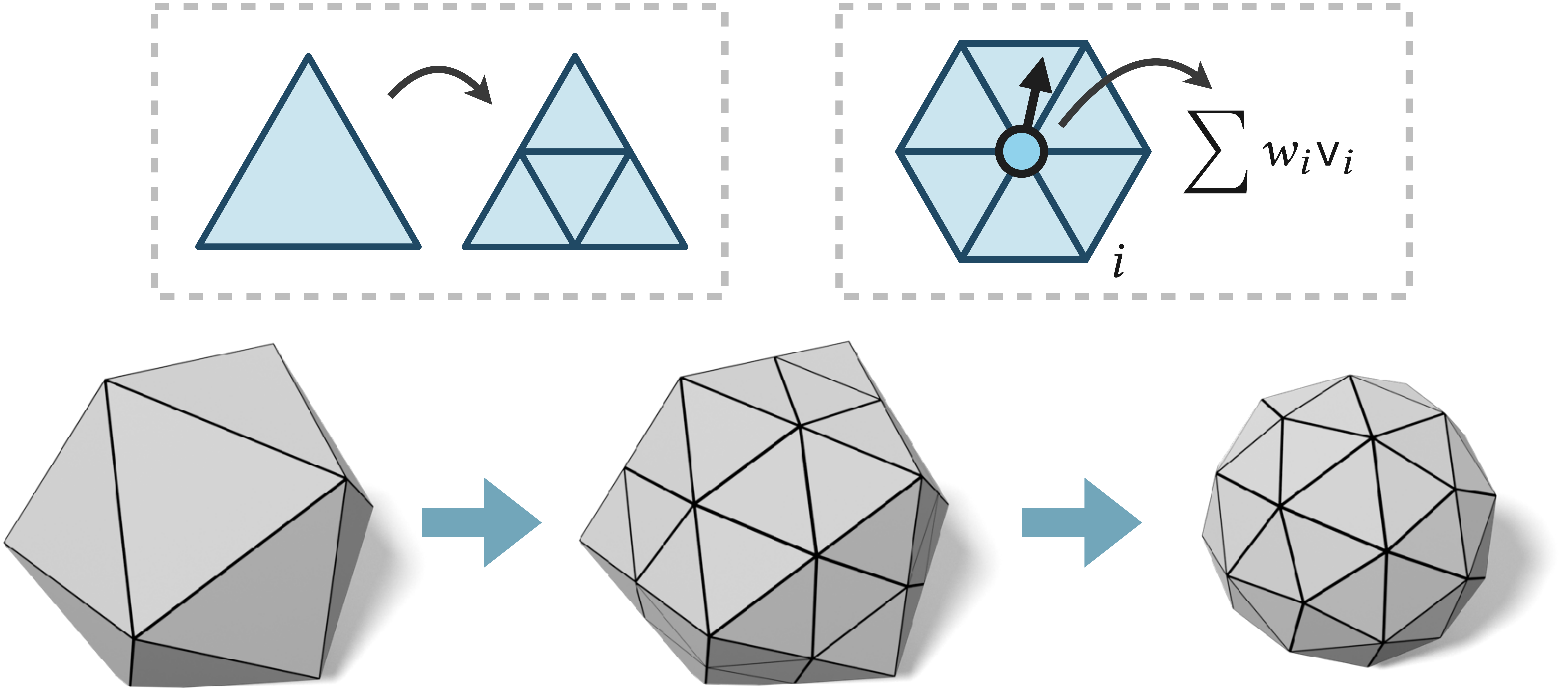




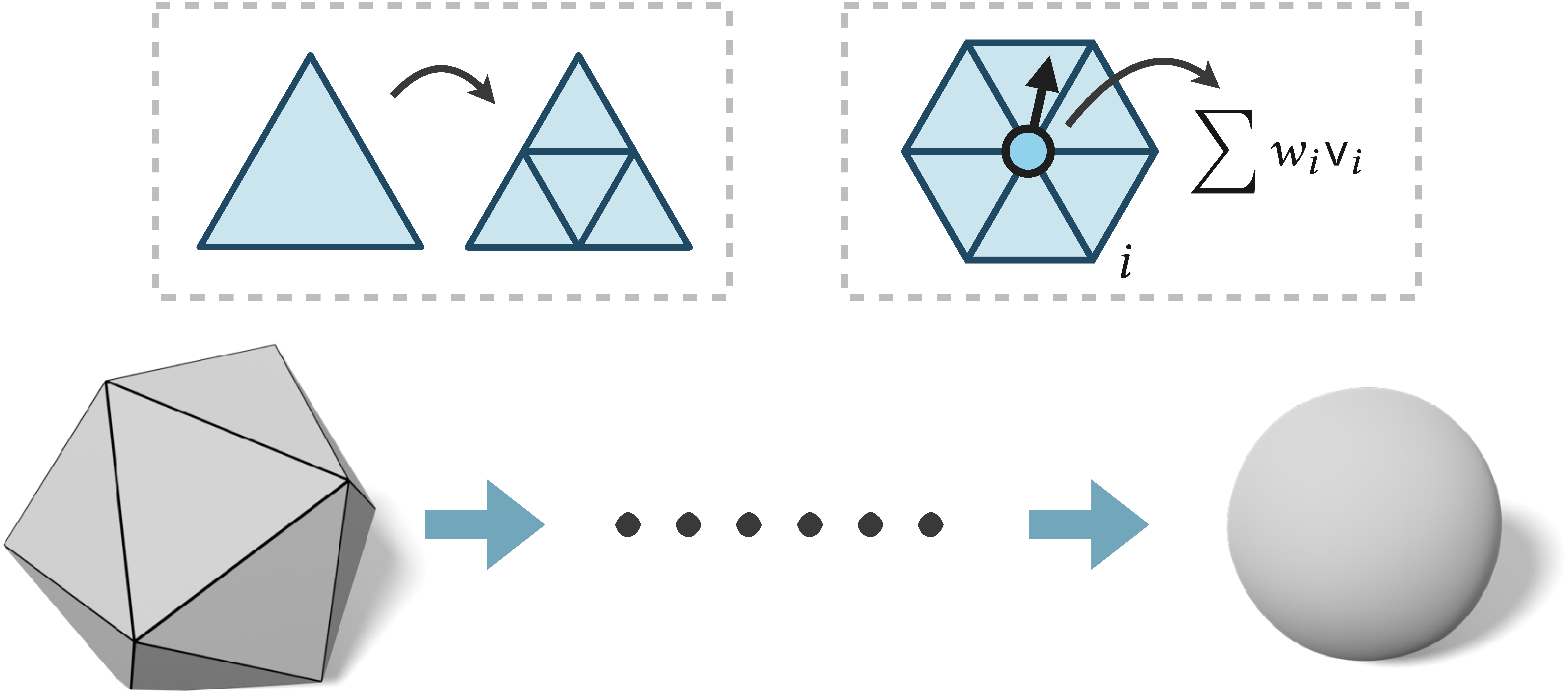
- ✗ assume 269 vertices
- ✗ assume a specific connectivity
- ✗ assume a specific ordering
- ✗ assume a specific orientation

- ✗ assume 1070 vertices
- ✗ assume a specific connectivity
- ✗ assume a specific ordering
- ✗ assume a specific orientation

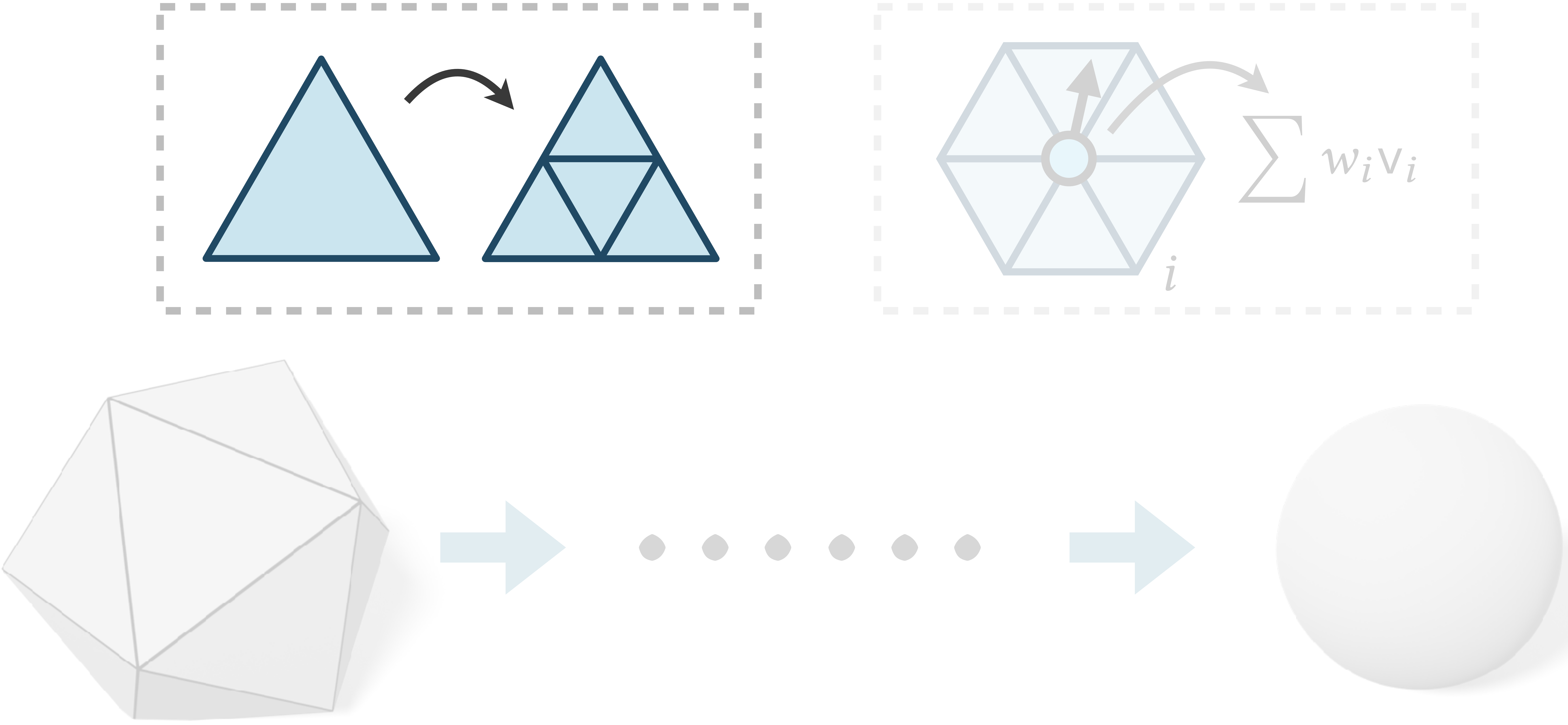
Classic Subdivision



Classic Subdivision



Where we should put the network?



Discrete Variables

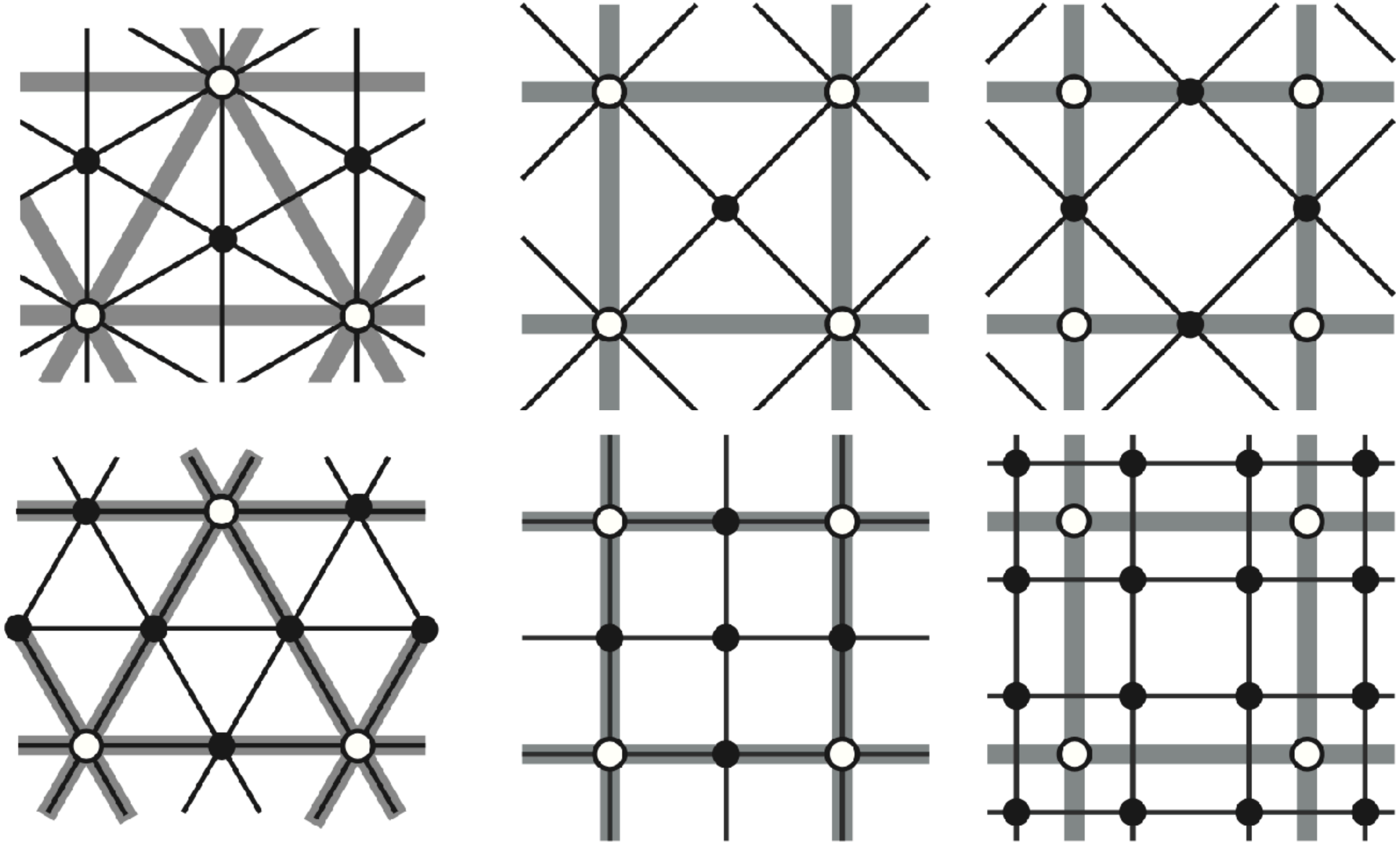
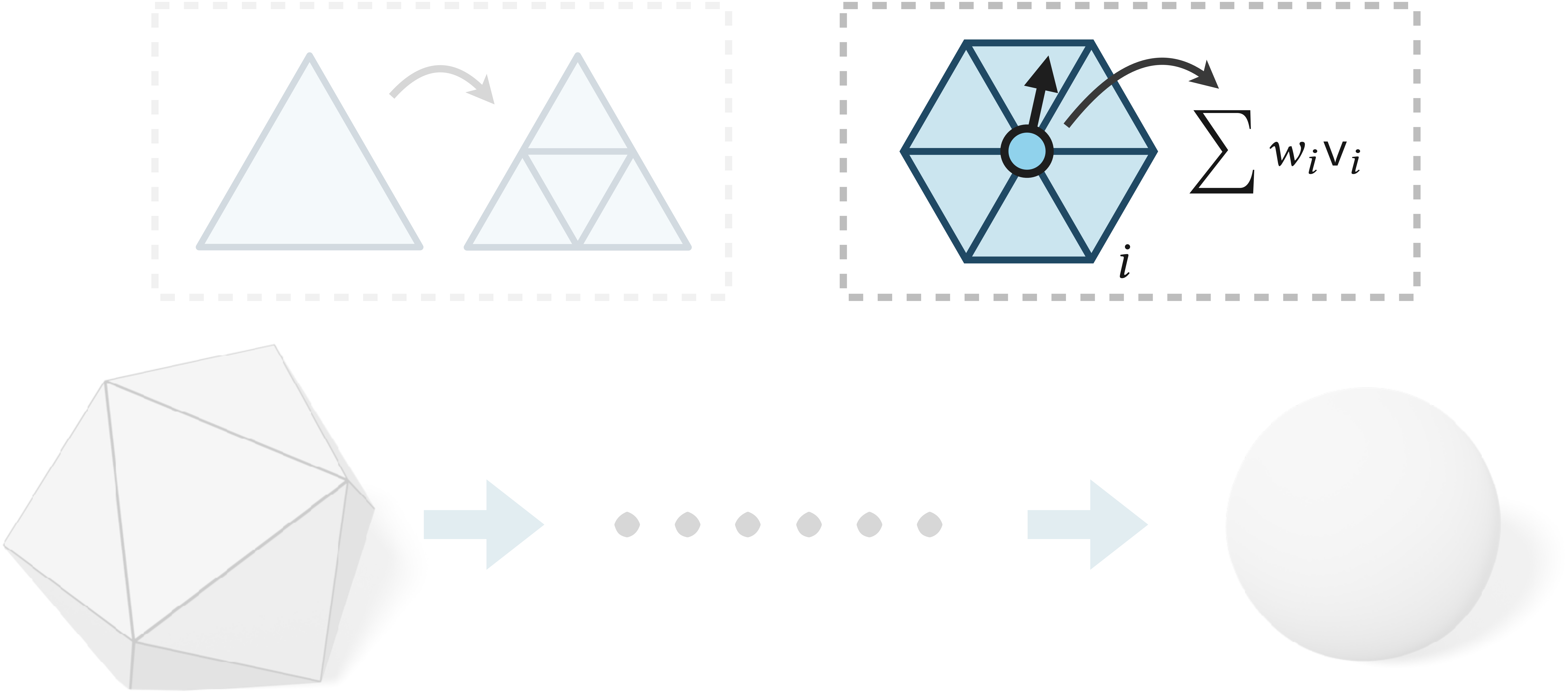


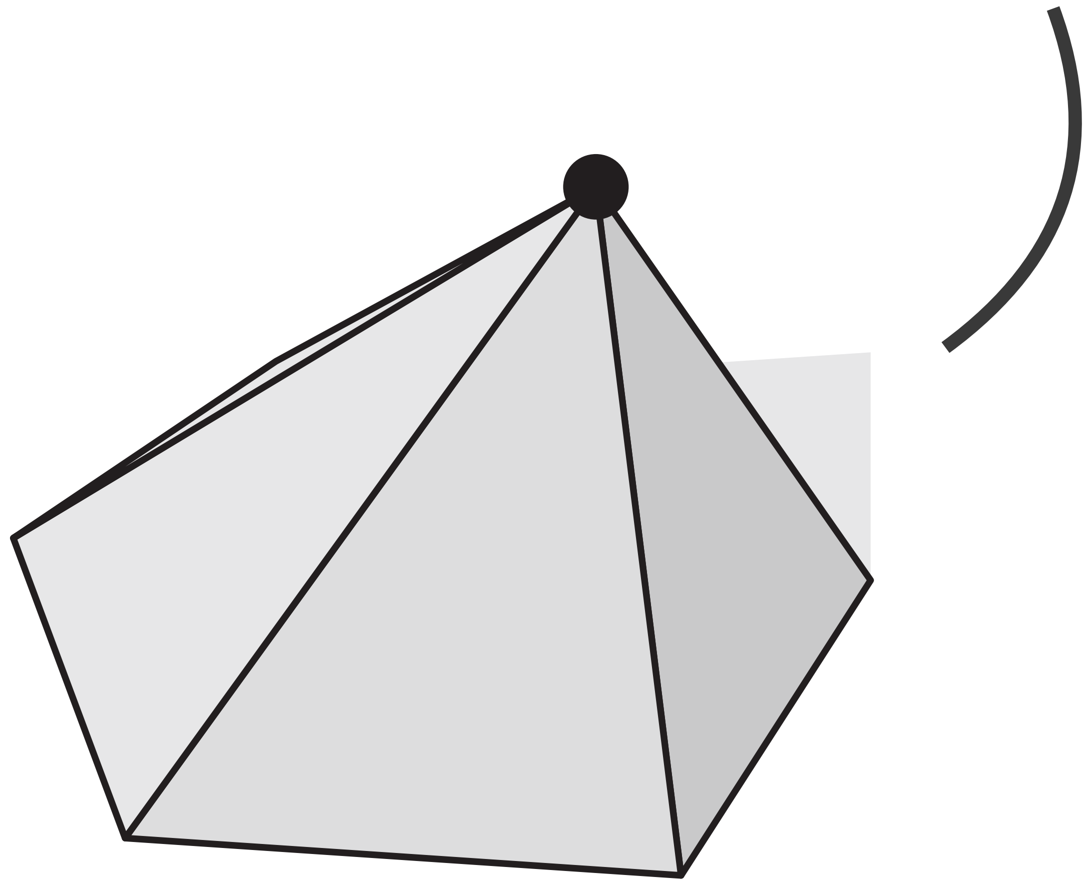
image source: Zorin 2005

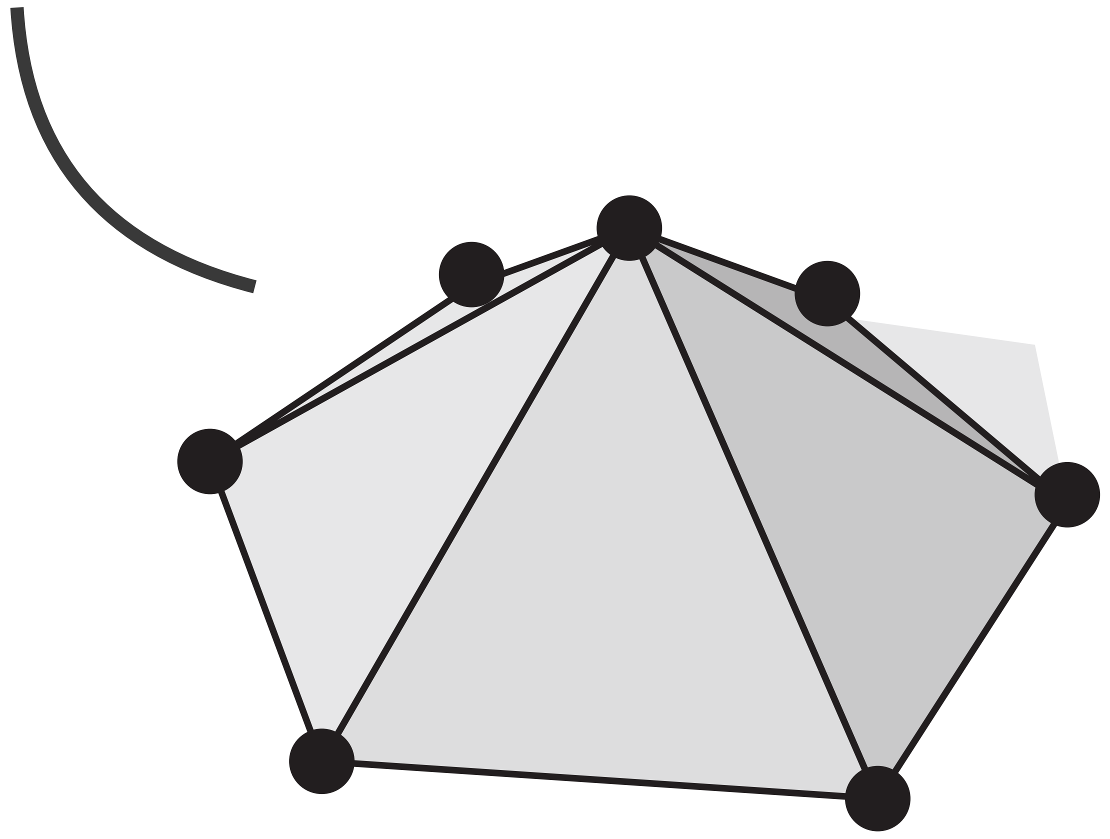
Where we should put the network?



From features to locations

$$y = f_{\theta}(x)$$

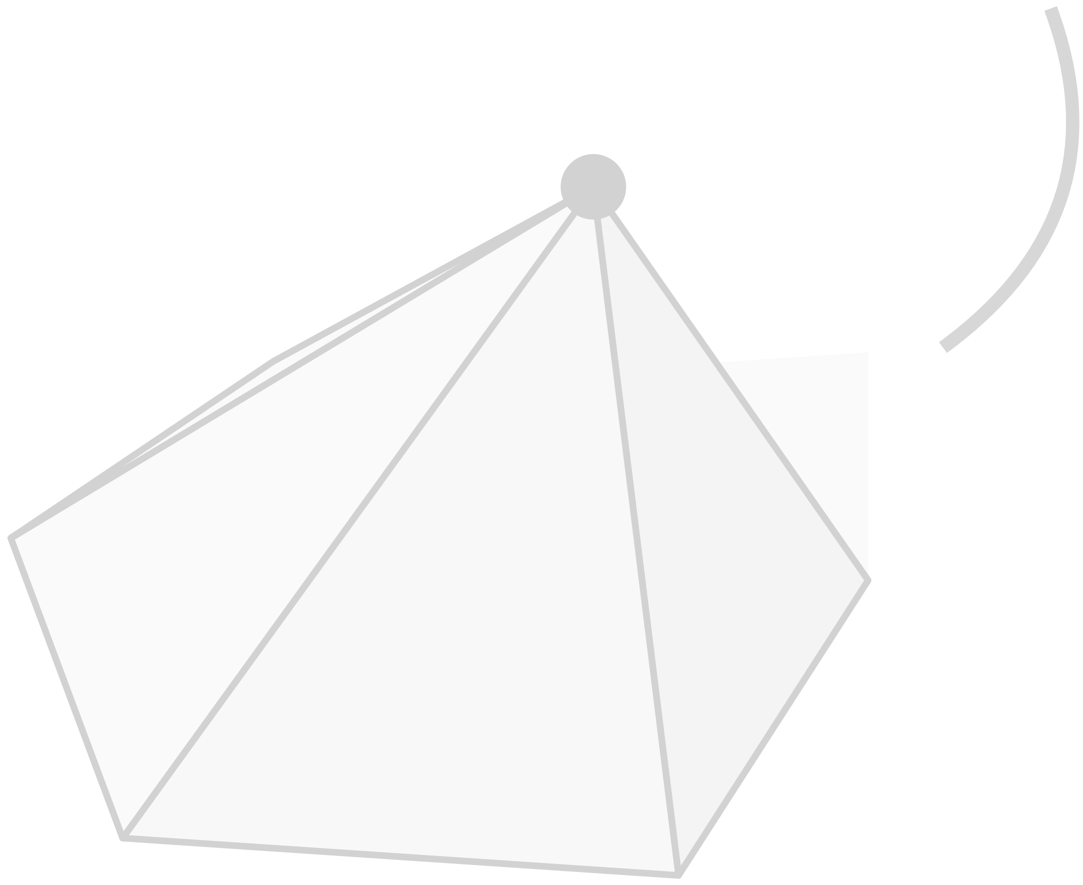


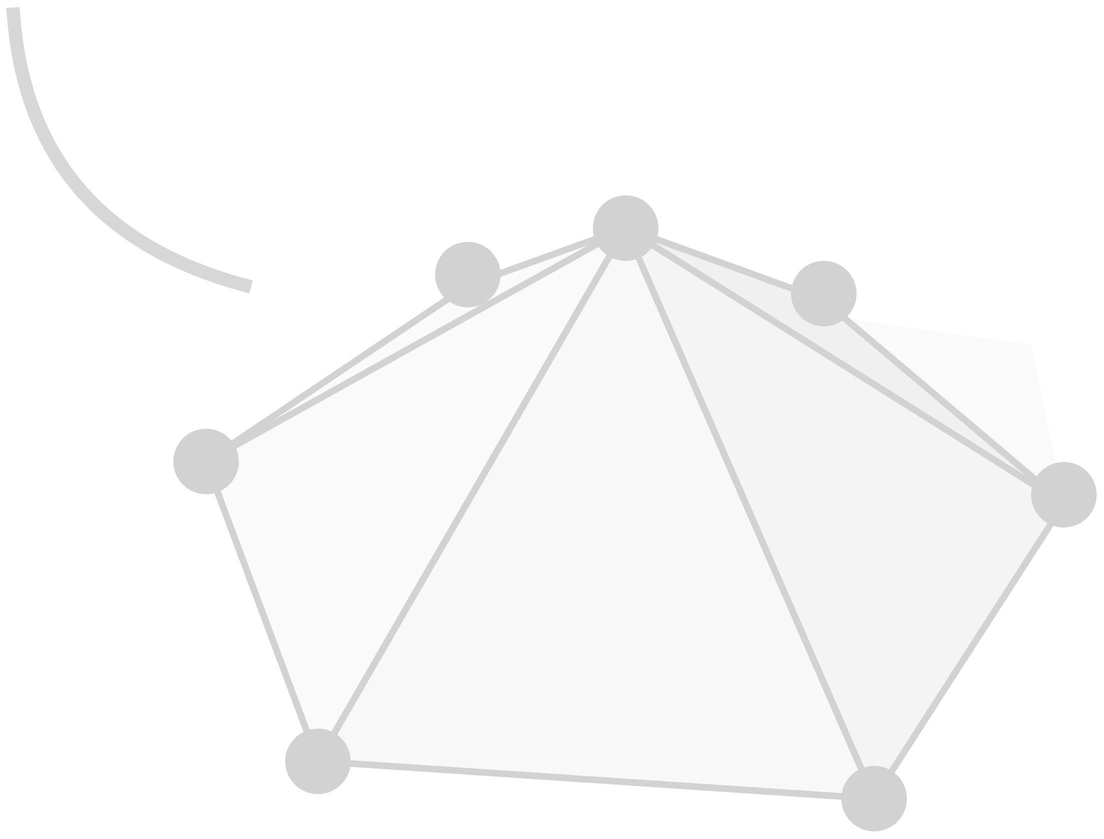


From features to locations

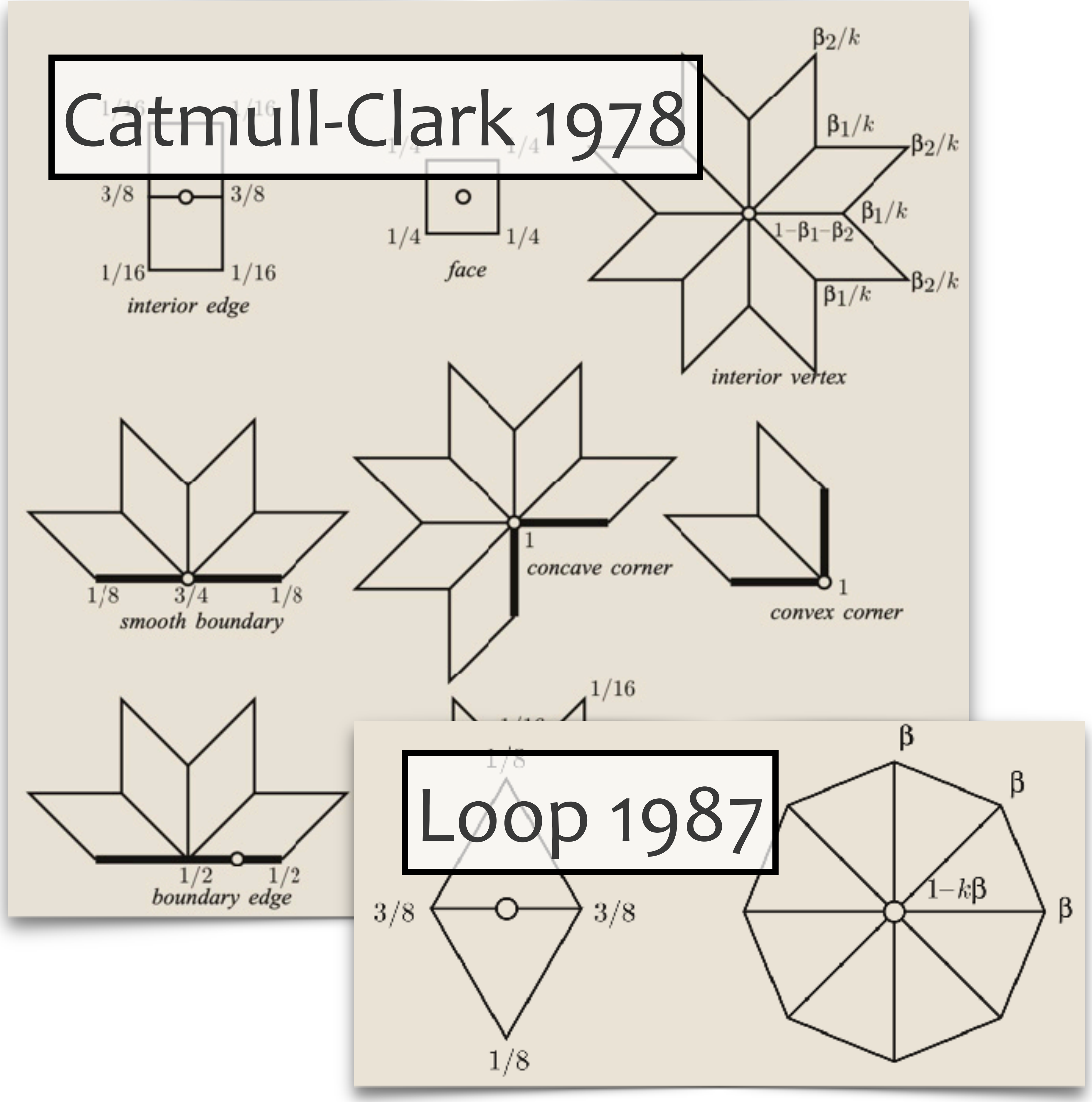
difficult to define

$$y = f_\theta(x)$$





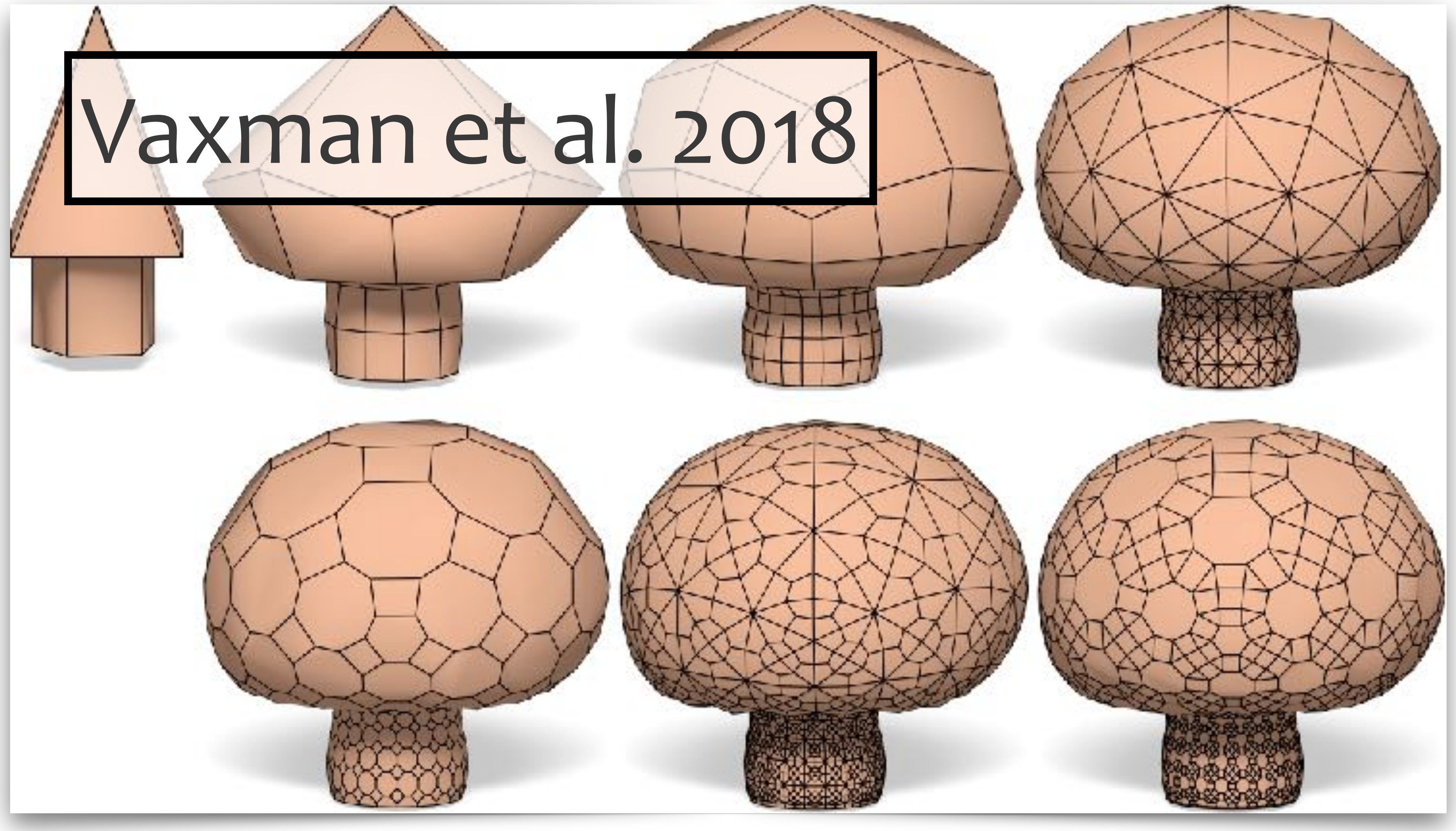
Still an ongoing research



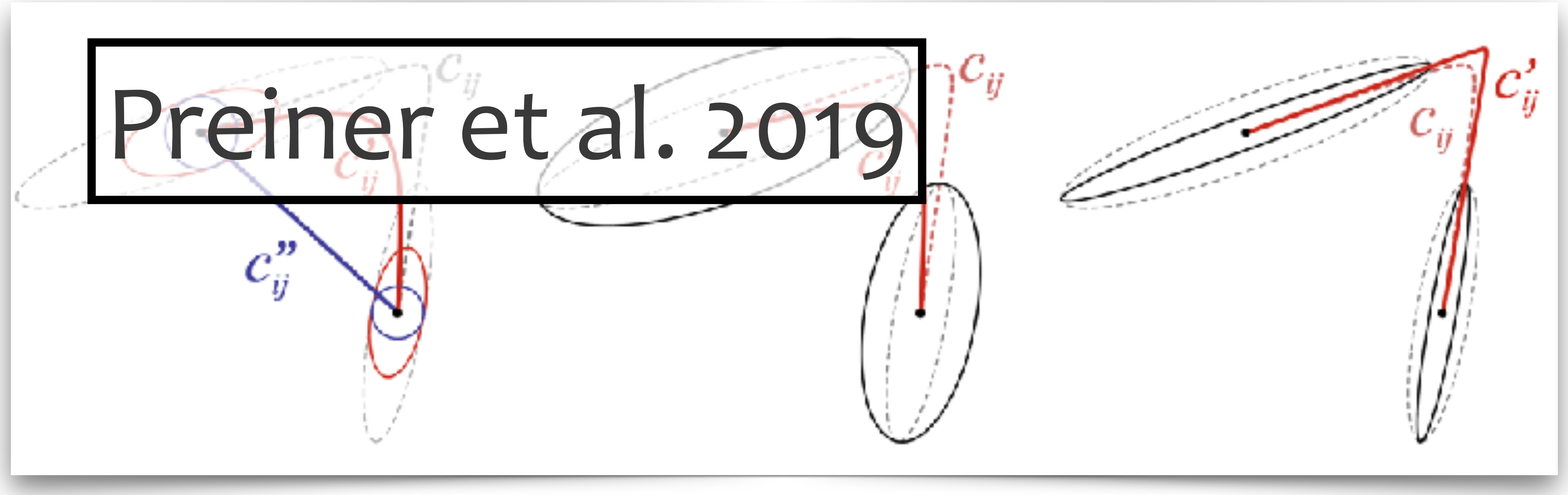
Catmull-Clark 1978

Loop 1987

• • •



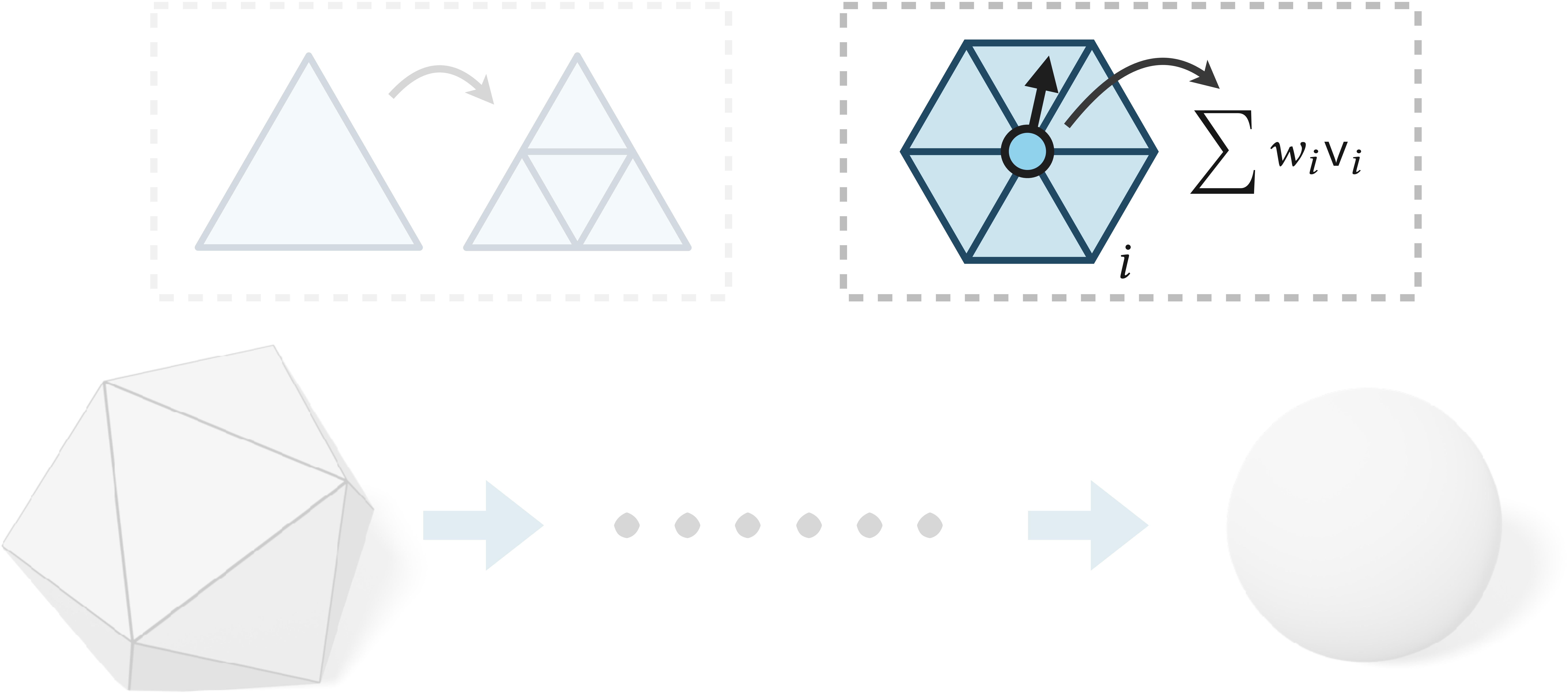
Vaxman et al. 2018



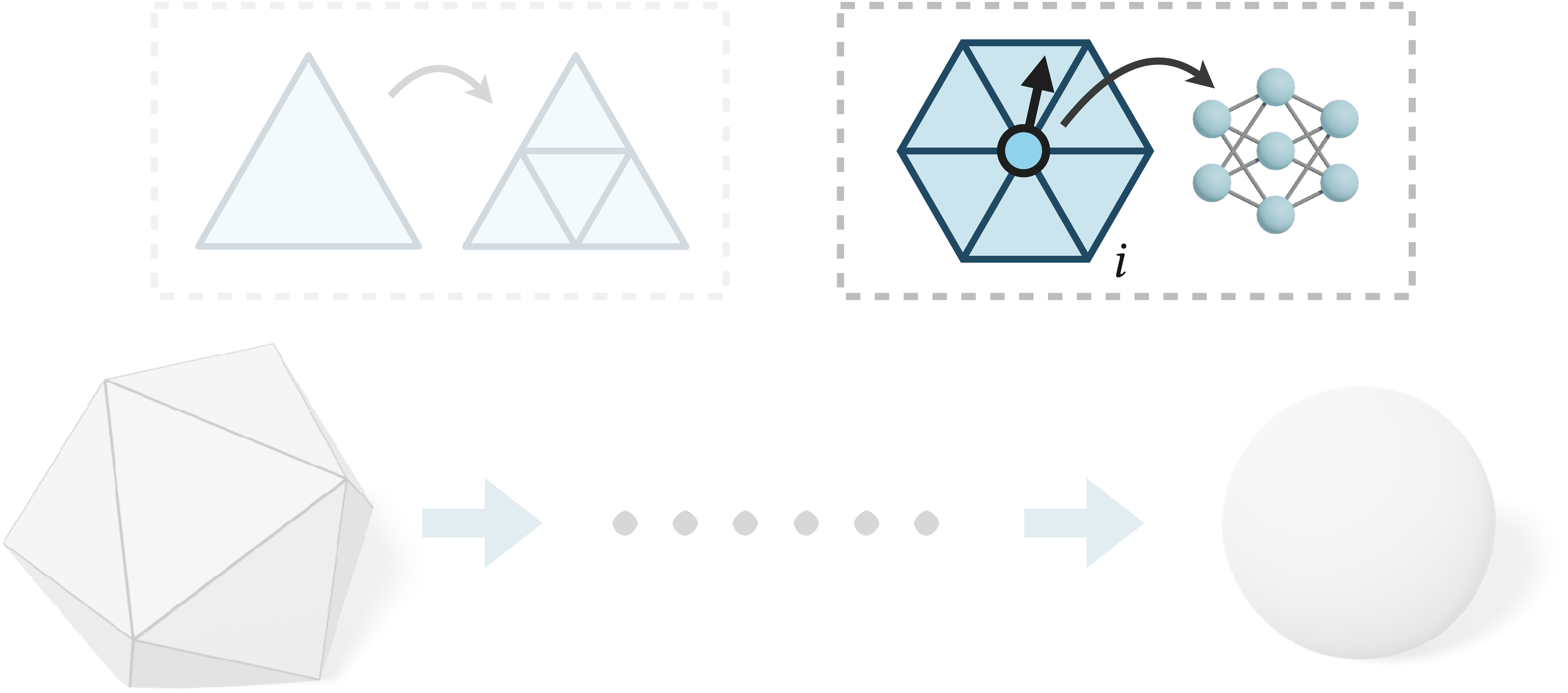
Preiner et al. 2019

image source: Zorin 2005

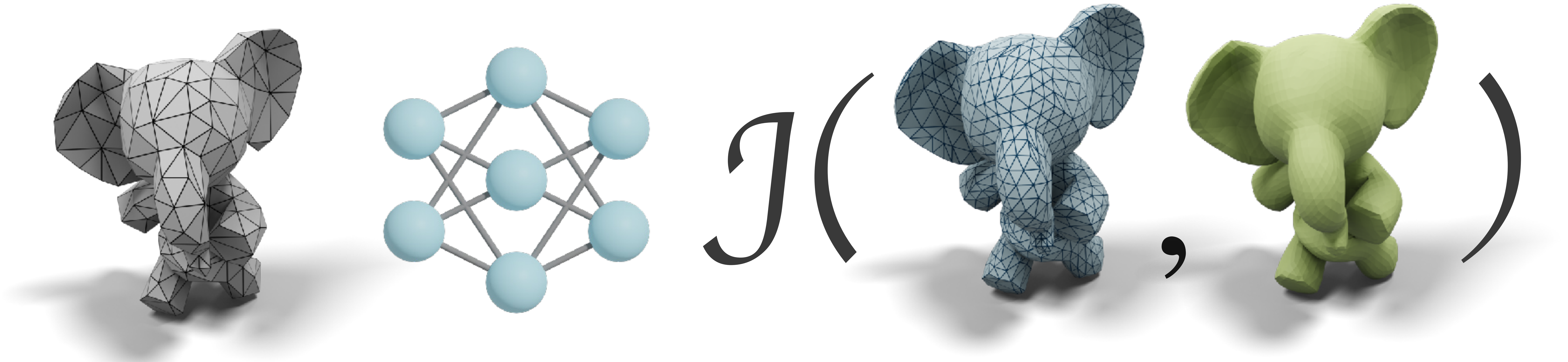
Where we should put the network?



Where we should put the network?



Neural Subdivision Training

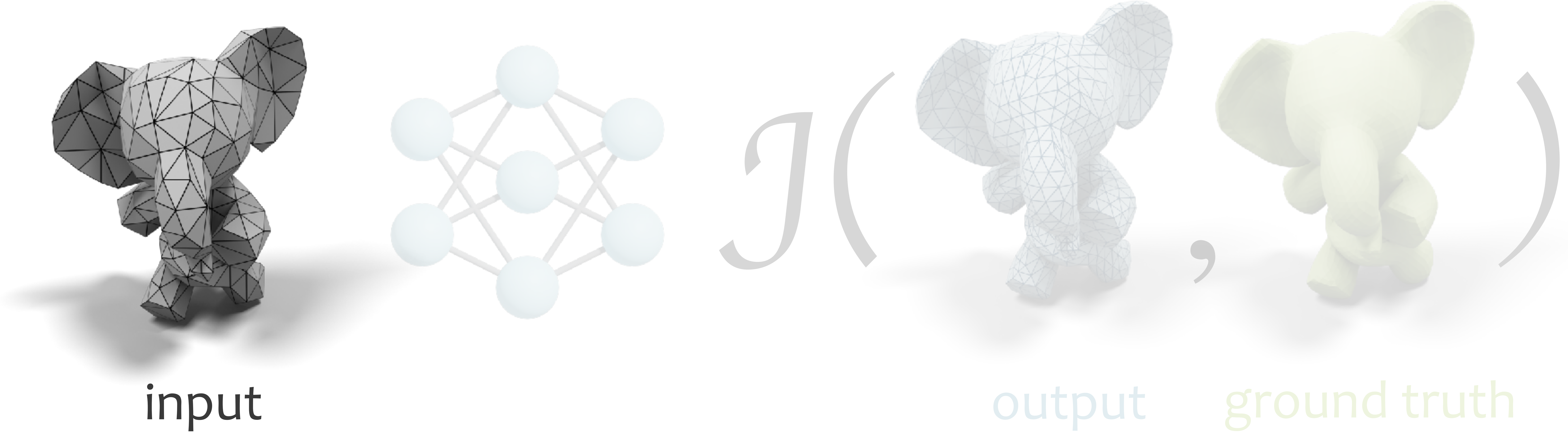


input

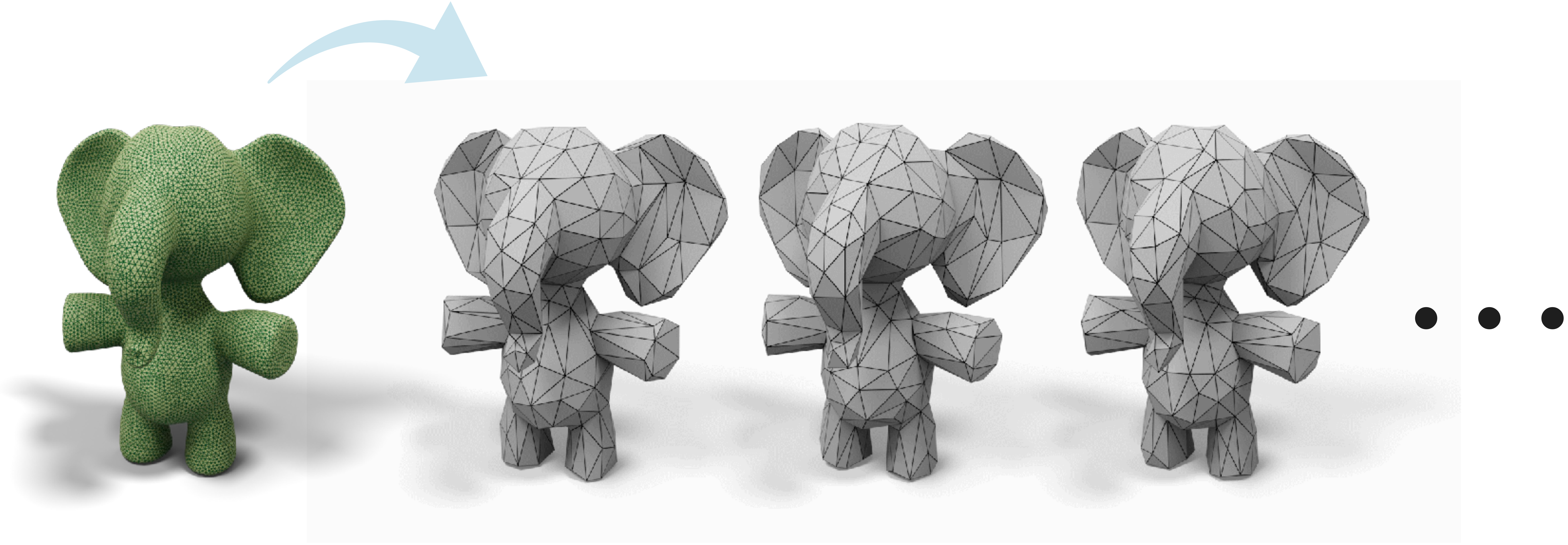
output

ground truth

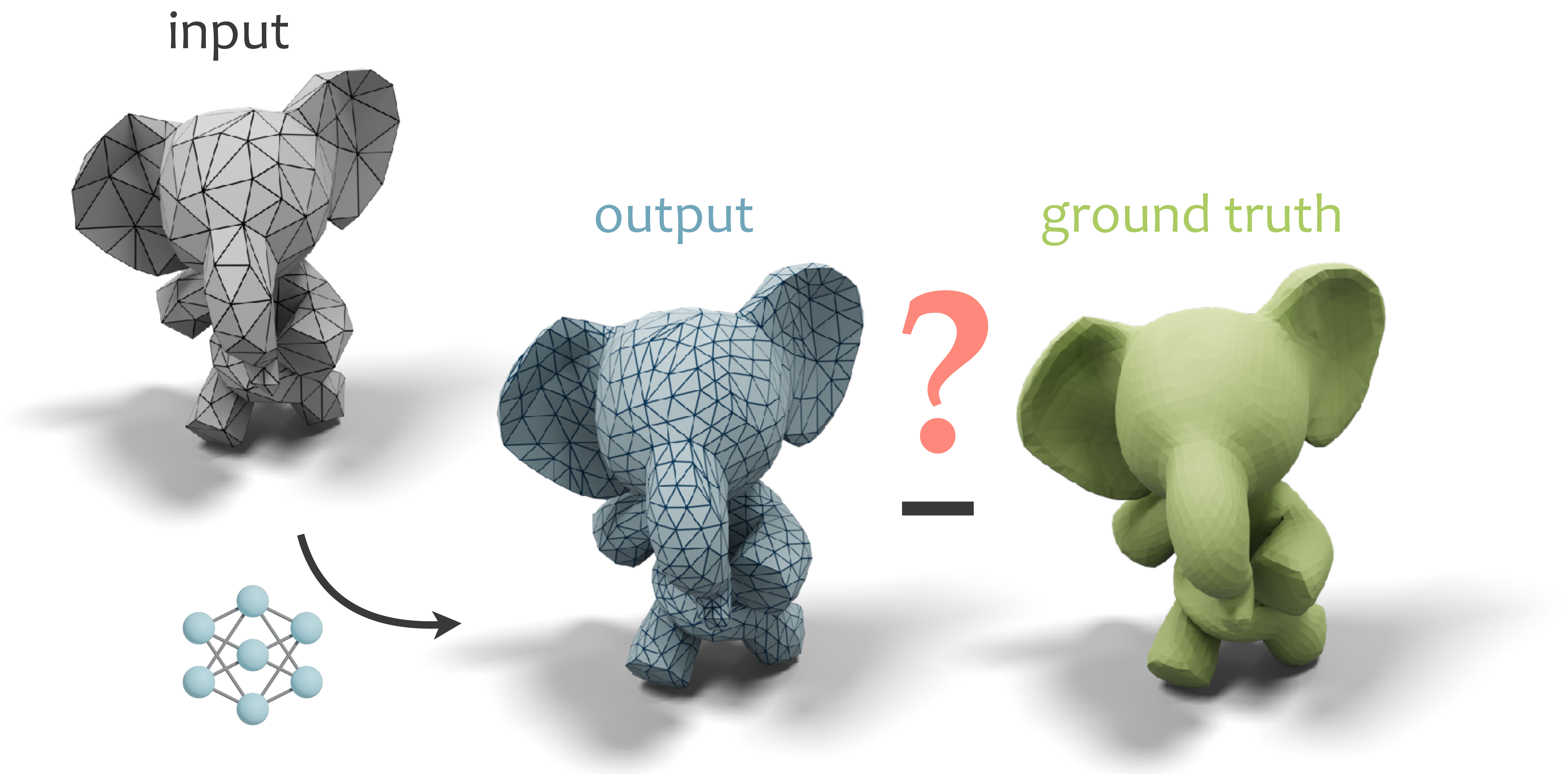
Neural Subdivision Training



Training Data



Neural Subdivision Training

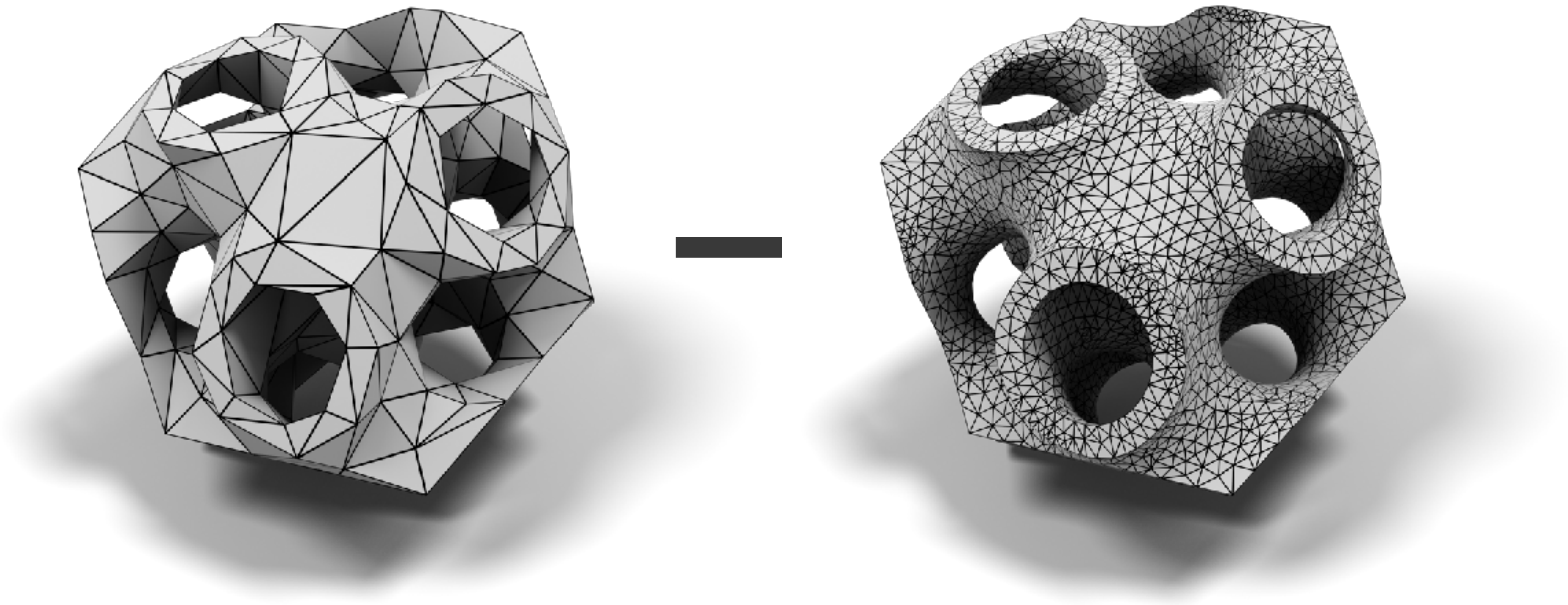


input

output

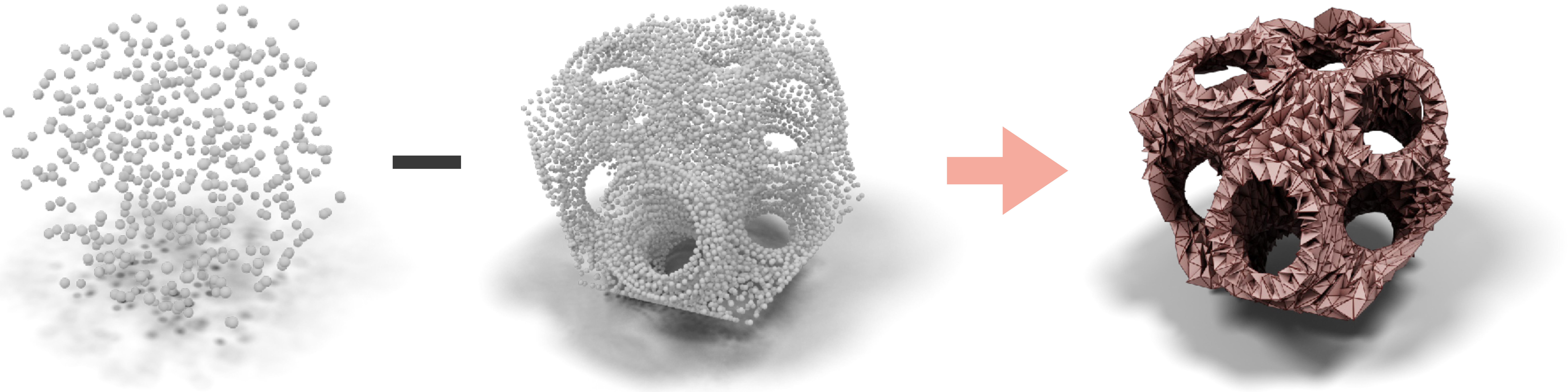
ground truth

Chamfer Distance



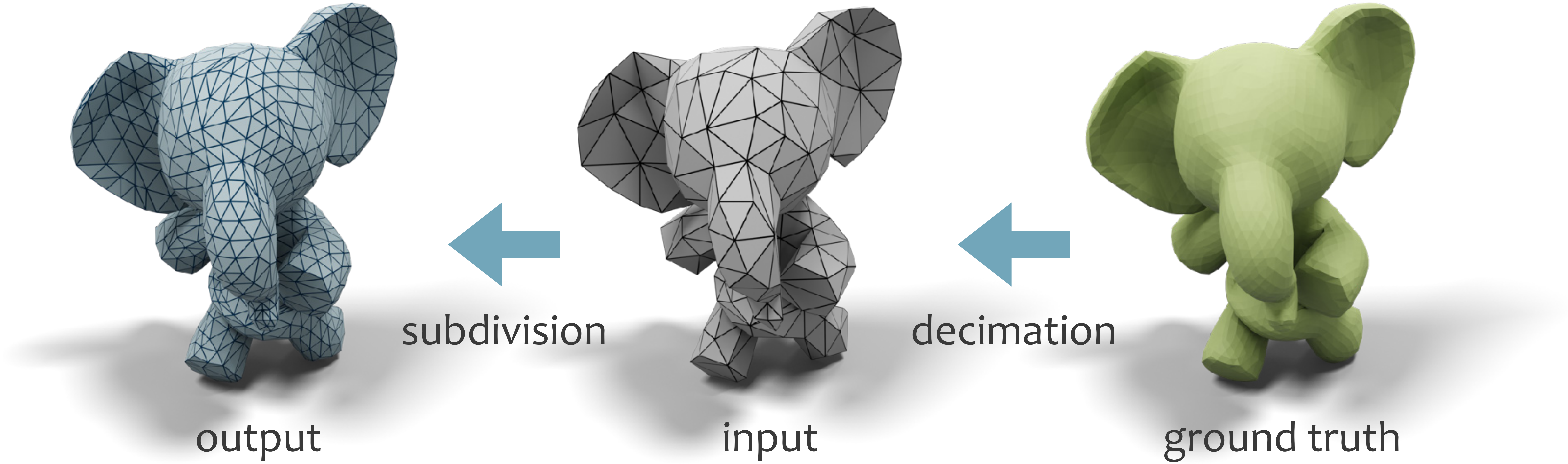
Barrow et. al. 1977, Fan et. al. 2017

Chamfer Distance



Barrow et. al. 1977, Fan et. al. 2017

Leverage the Structure



subdivision

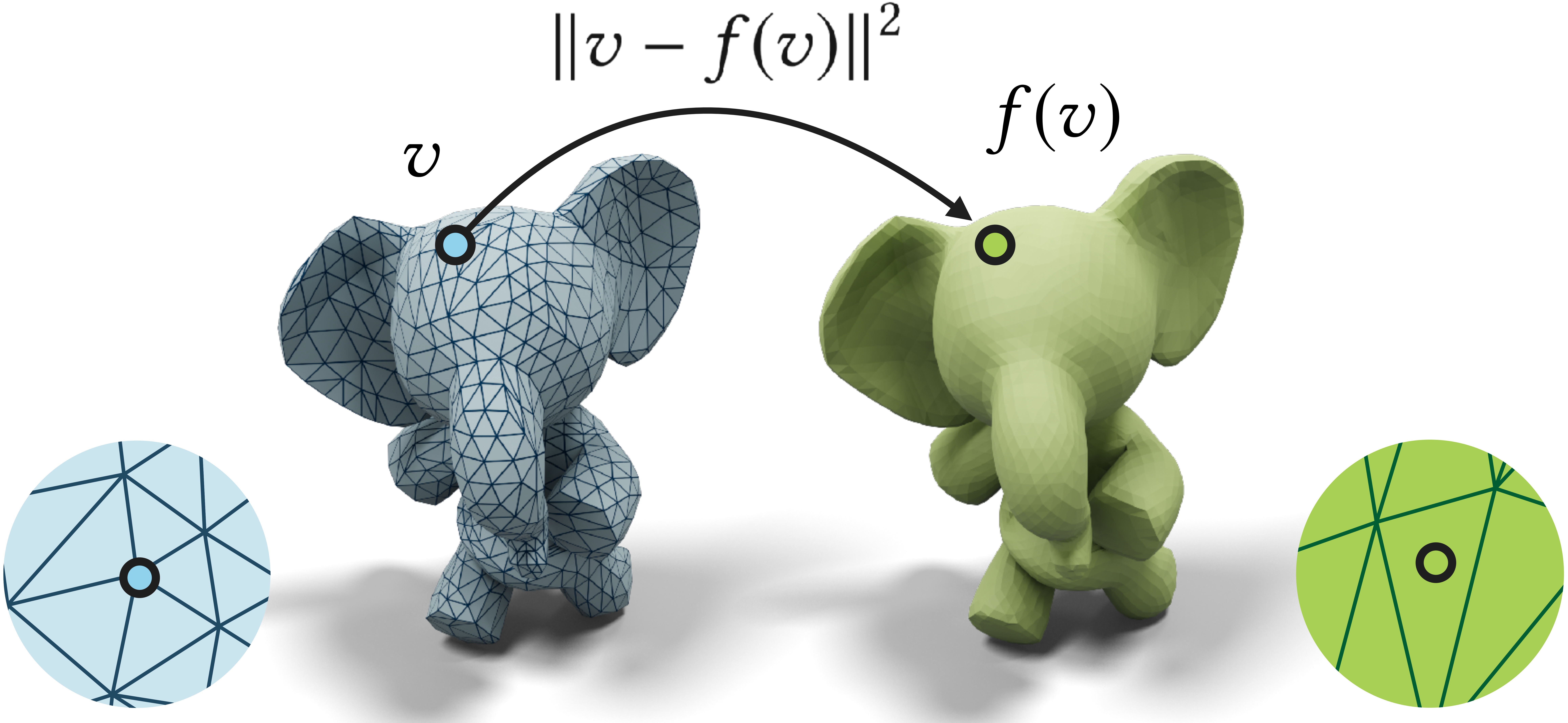
decimation

output

input

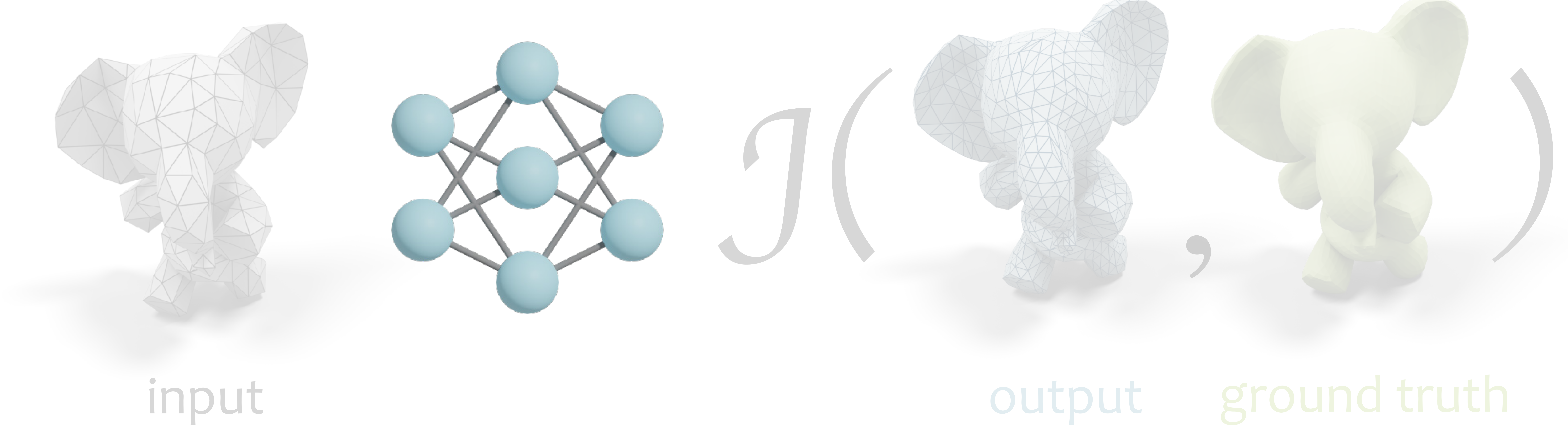
ground truth

Bijective Mapping

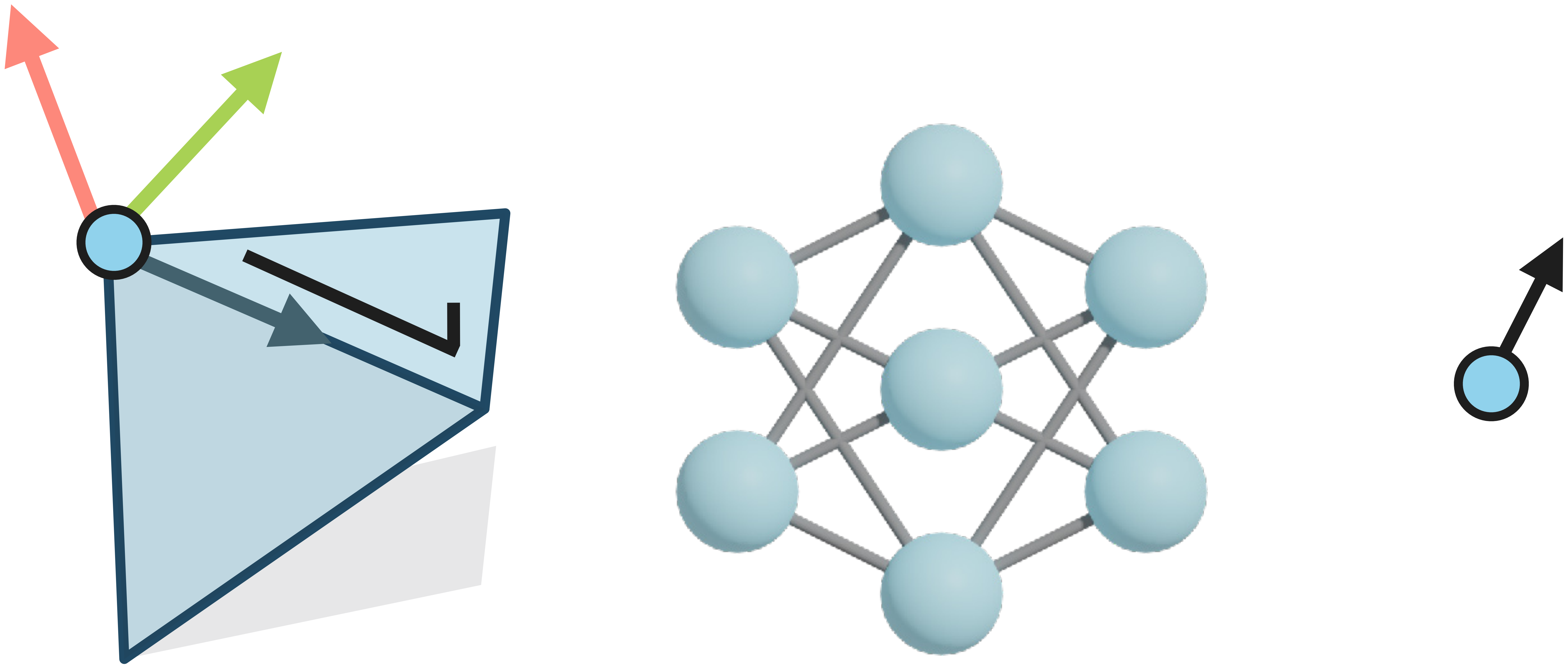


$$\|v - f(v)\|^2$$

Neural Subdivision Training



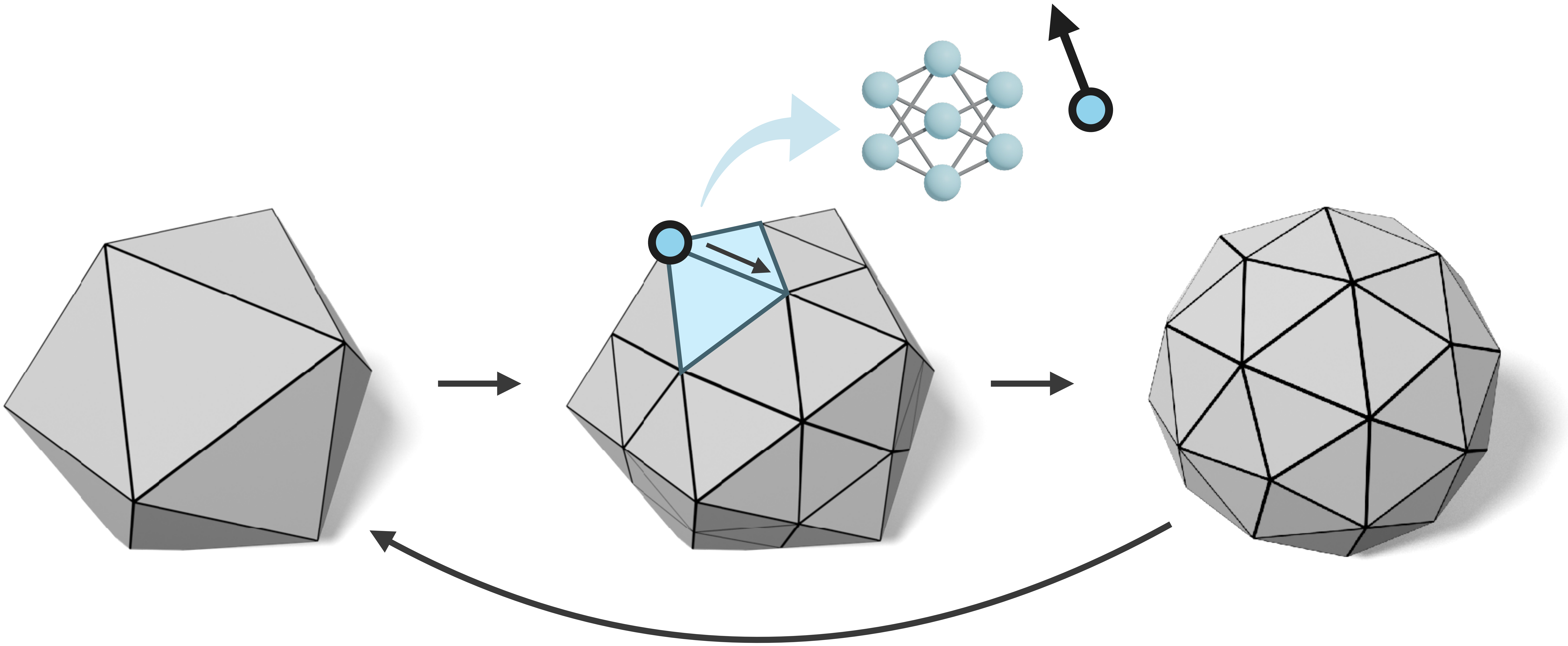
Subdivision Network



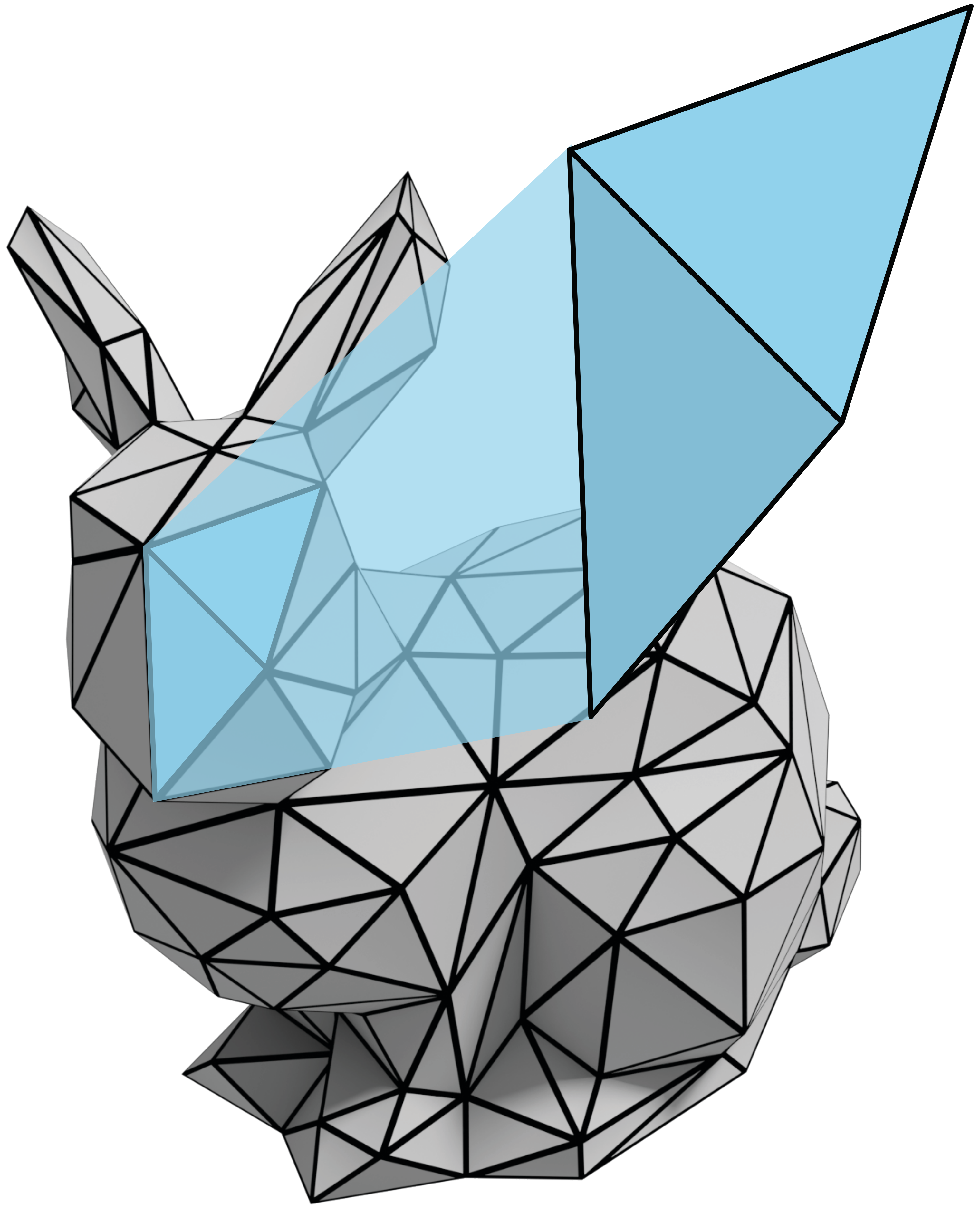
half-flap

displacement

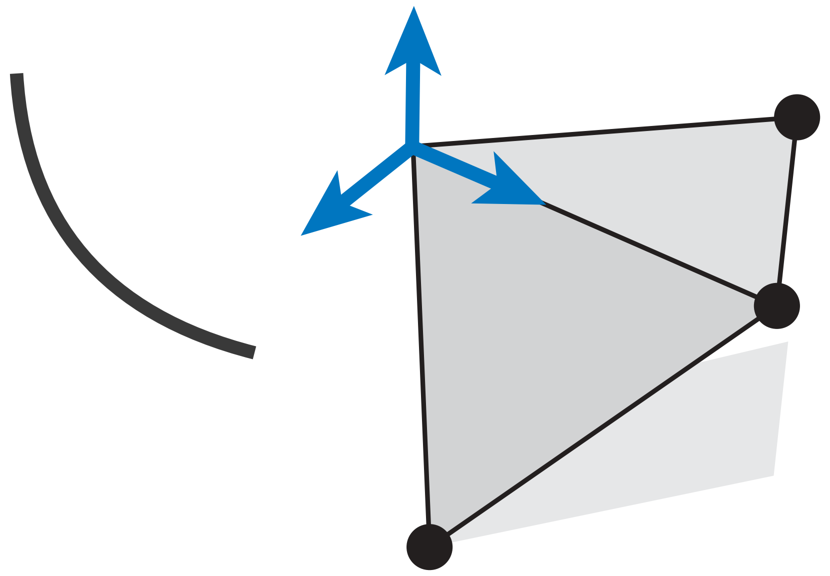
Subdivision Network



Key Takeaways



- ~~assume 1070 vertices~~
- ~~assume a specific connectivity~~
- ~~assume a specific ordering~~
- ~~assume a specific orientation~~



Alleviate limitations

quality

expensive training

generalization

large training data

push the limits

new possibilities

Alleviate limitations

quality

expensive training

generalization

large training data

push the limits

new possibilities

single shape training

Alleviate limitations

quality

expensive training

generalization

large training data

push the limits

new possibilities

single shape training

Alleviate limitations

quality

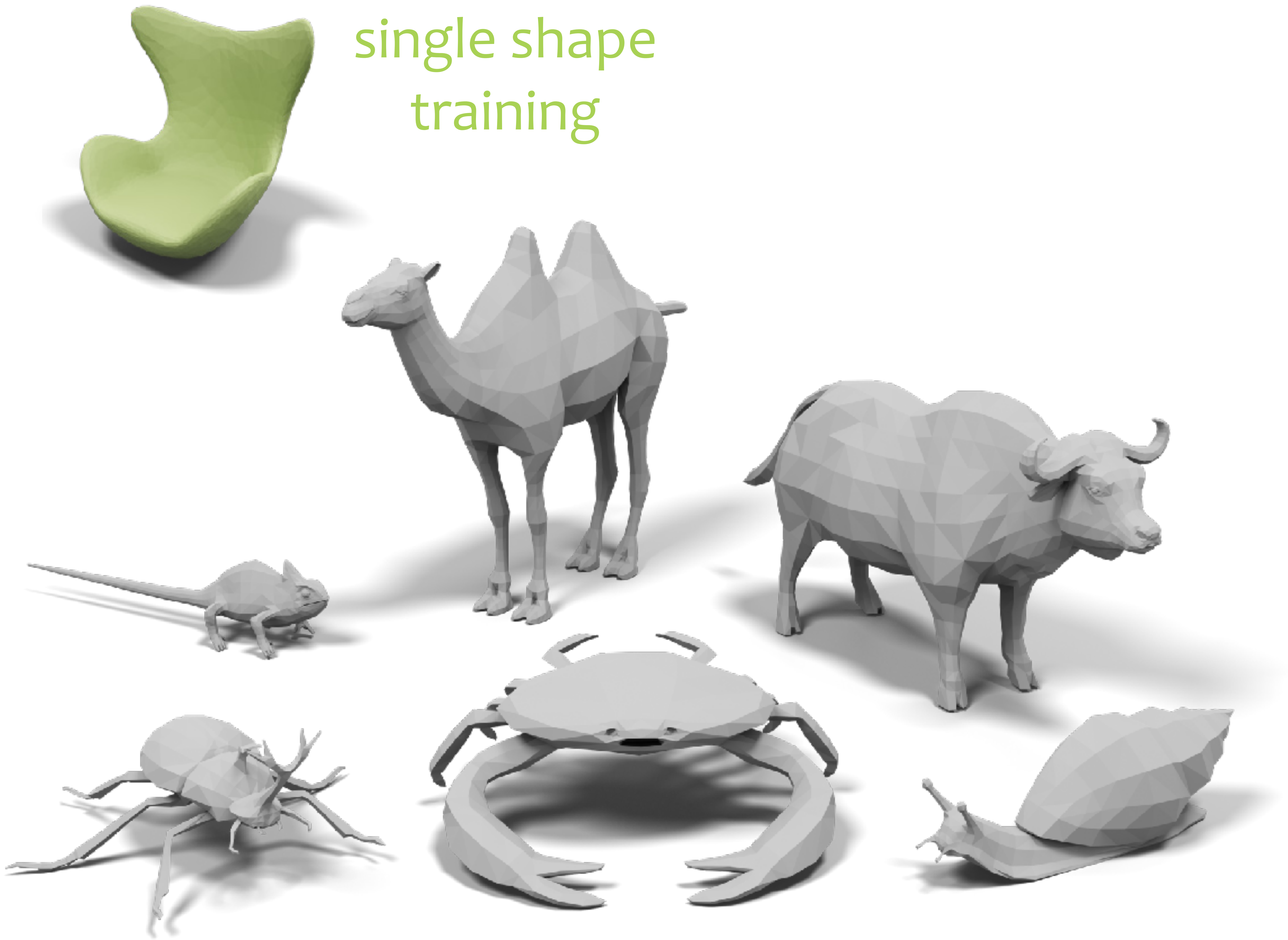
expensive training

generalization

large training data

push the limits

new possibilities



single shape training

Alleviate limitations

quality

expensive training

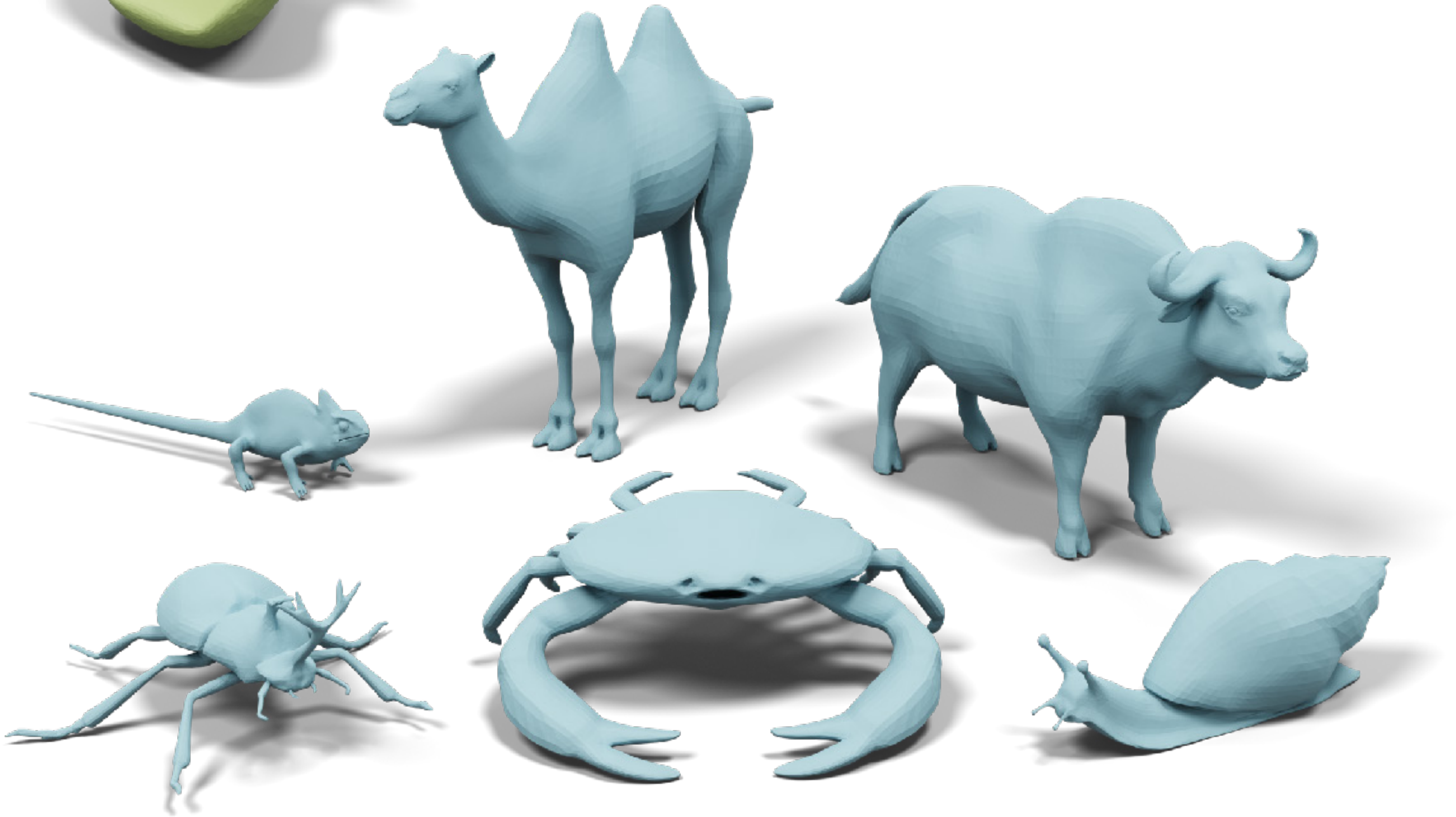
generalization

large training data

push the limits

new possibilities

single shape training



Alleviate limitations

quality

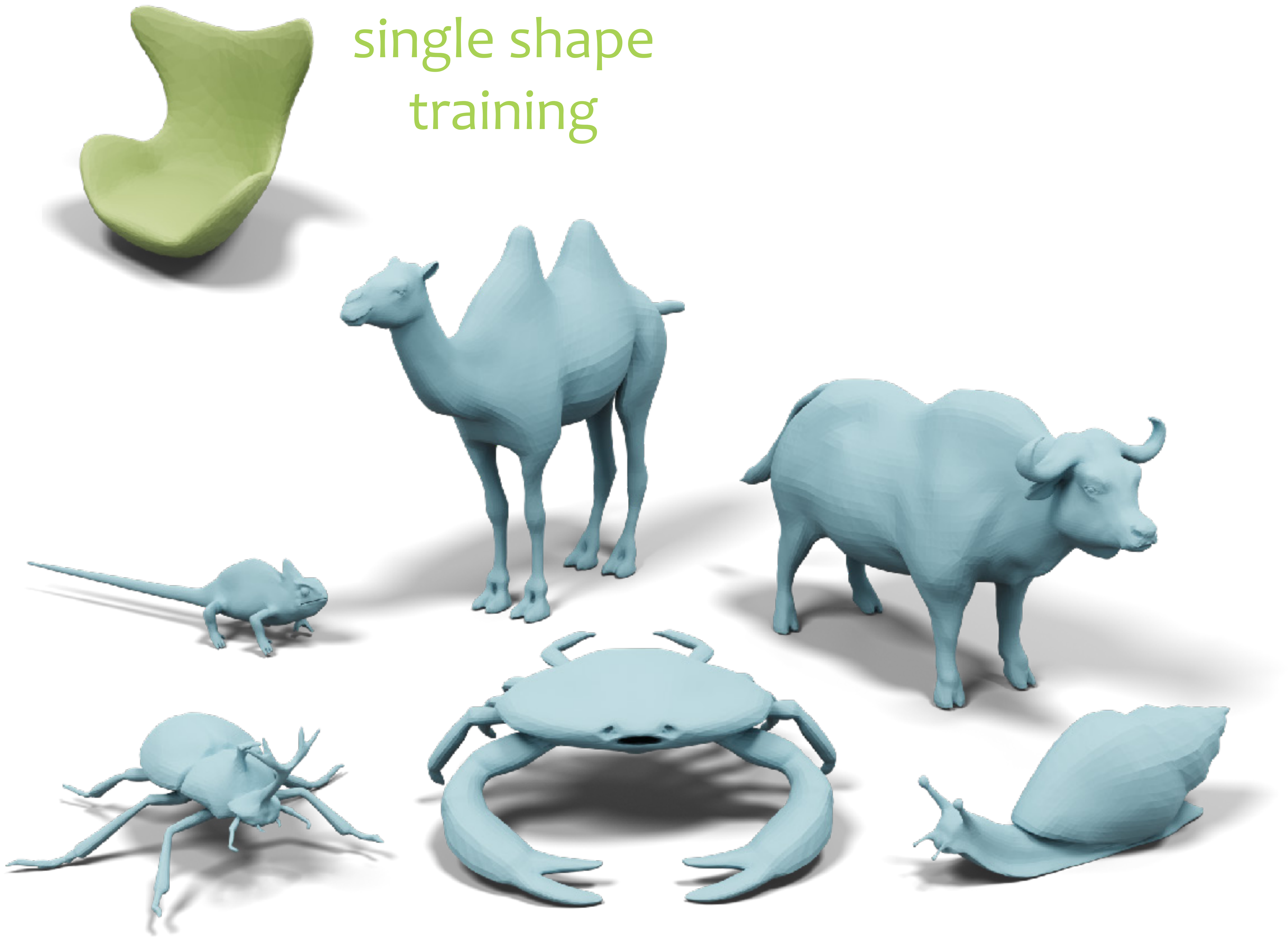
expensive training

generalization

large training data

push the limits

new possibilities



Enjoy Advantages

quality

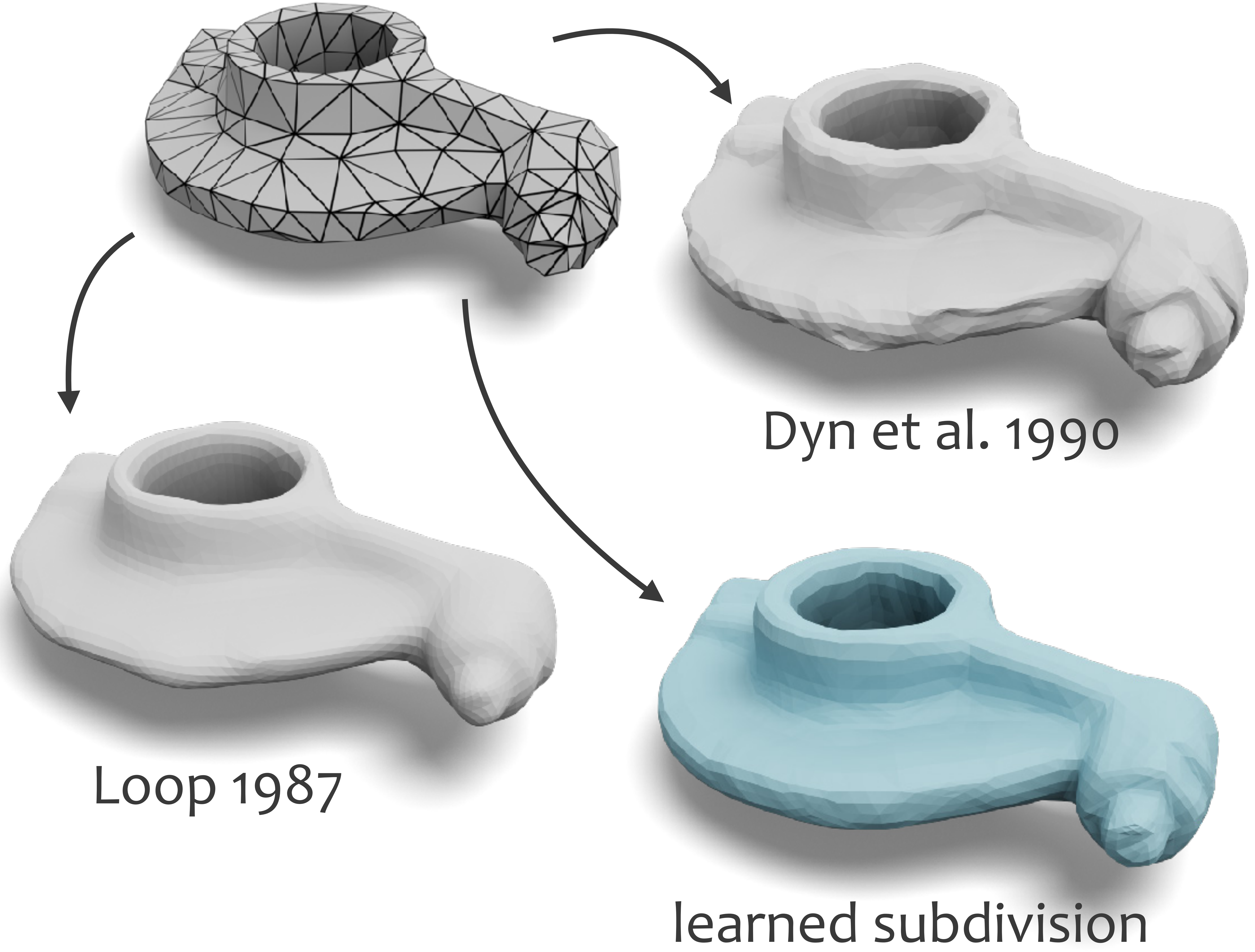
expensive training

generalization

large training data

push the limits

new possibilities



Enjoy Advantages

quality

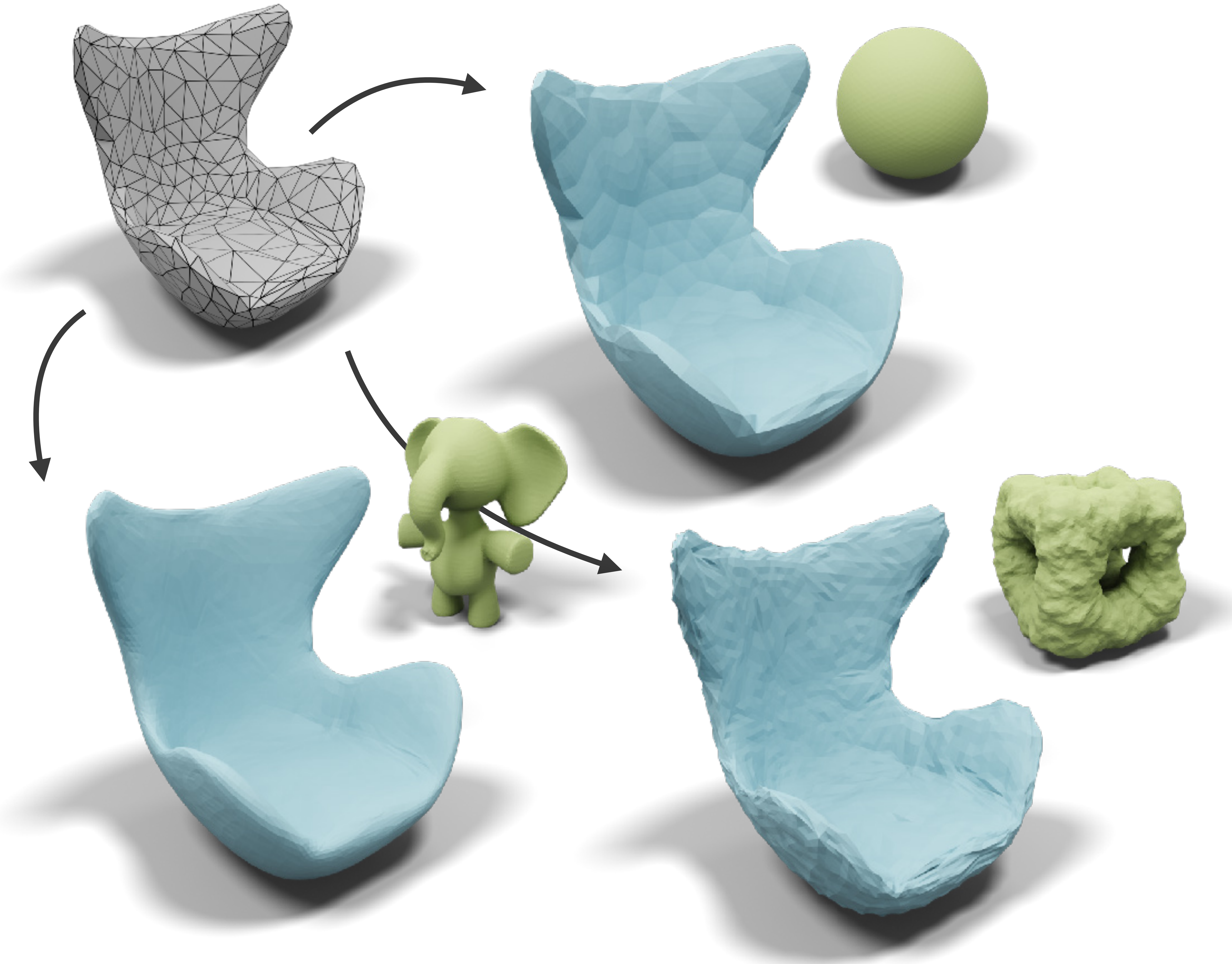
expensive training

generalization

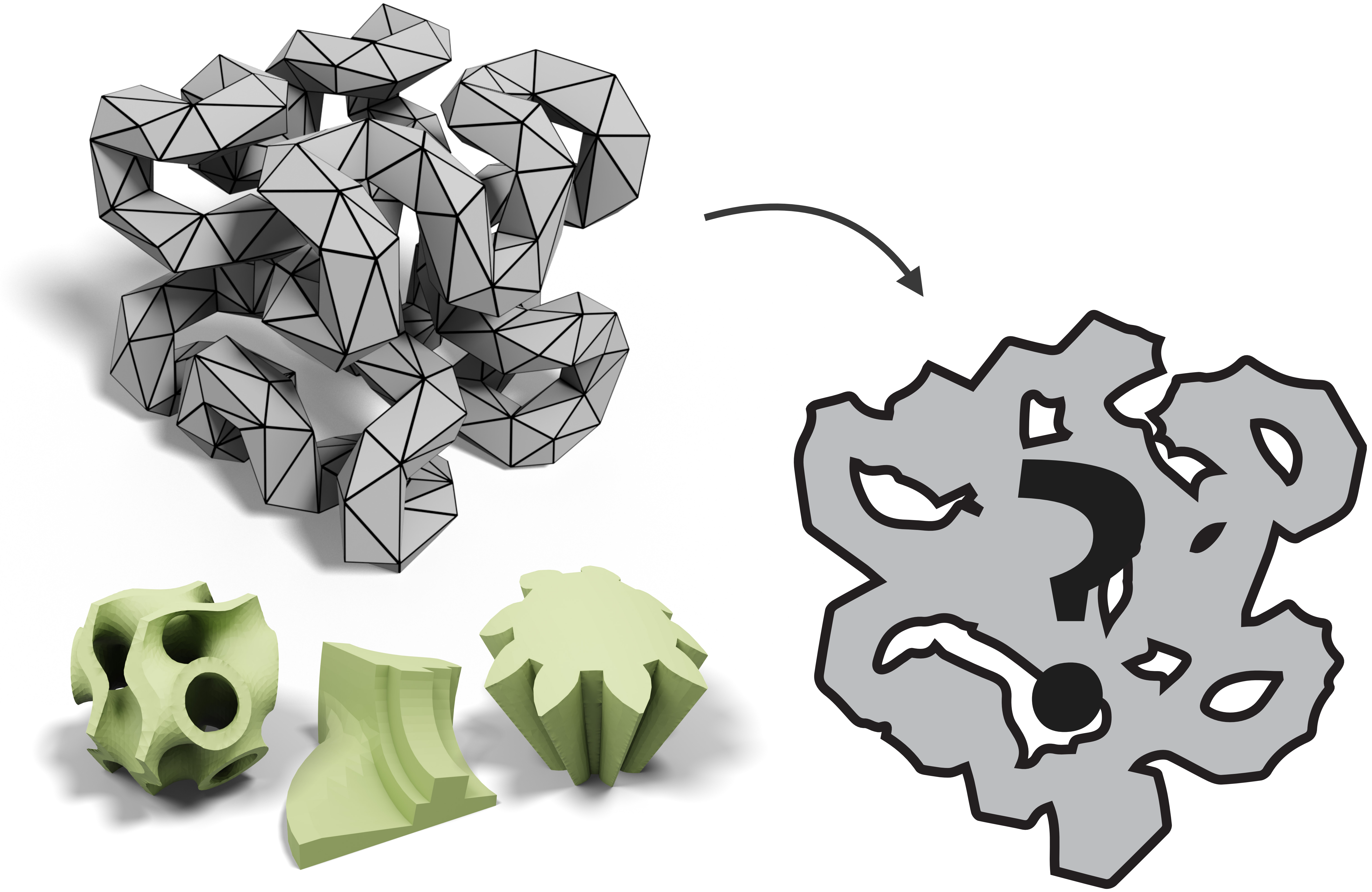
large training data

push the limits

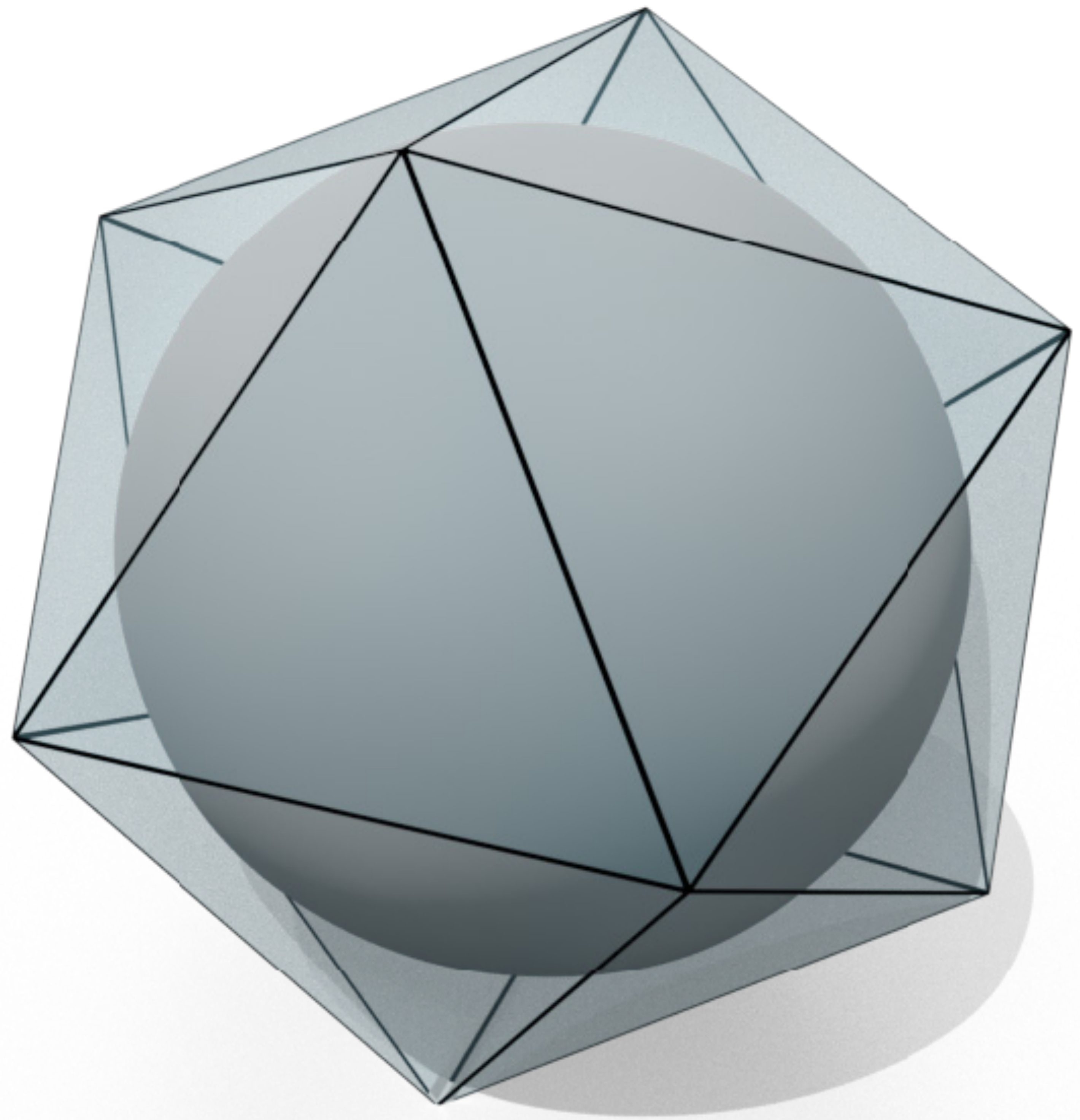
new possibilities



Ongoing Research

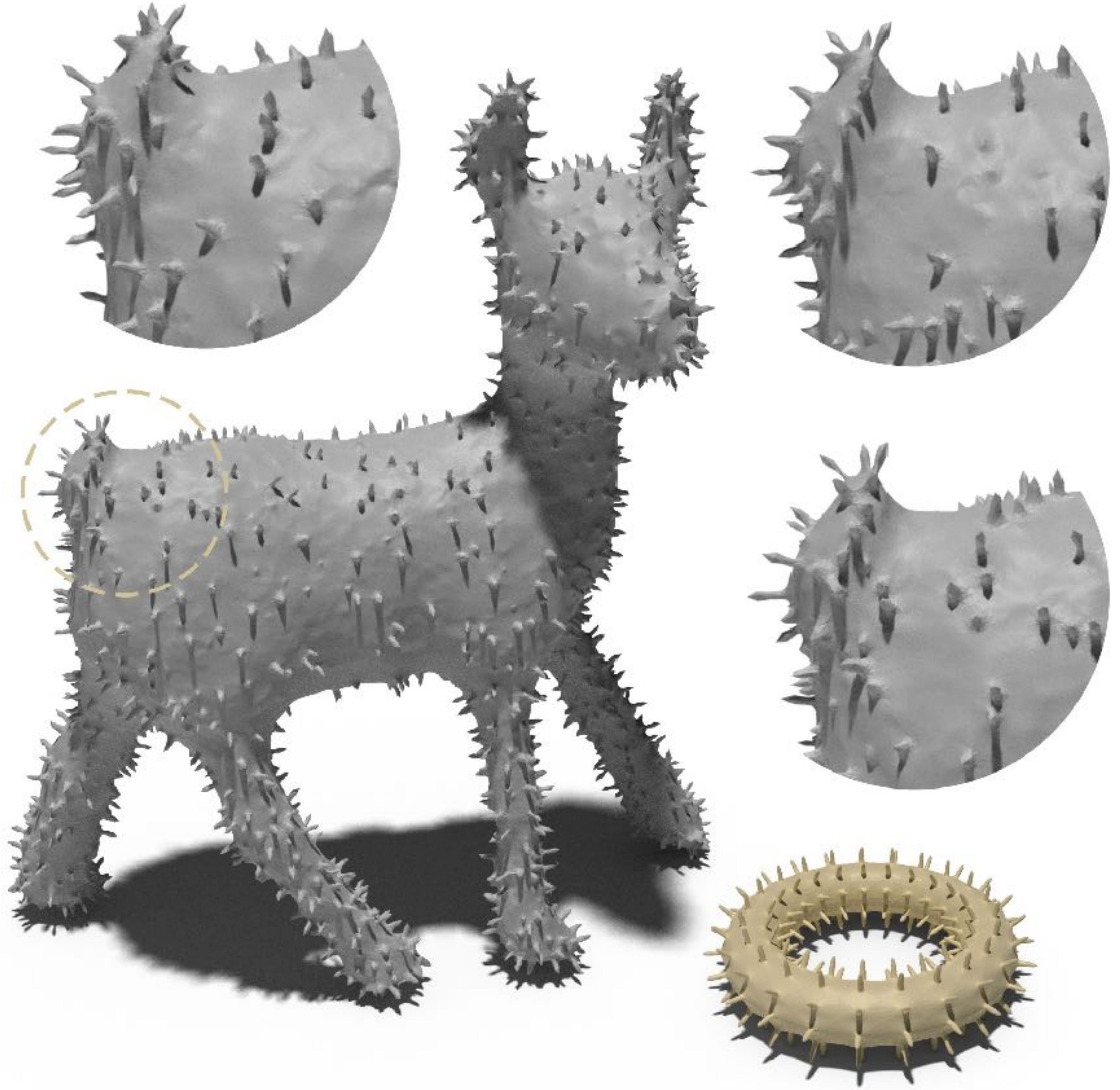


black-box

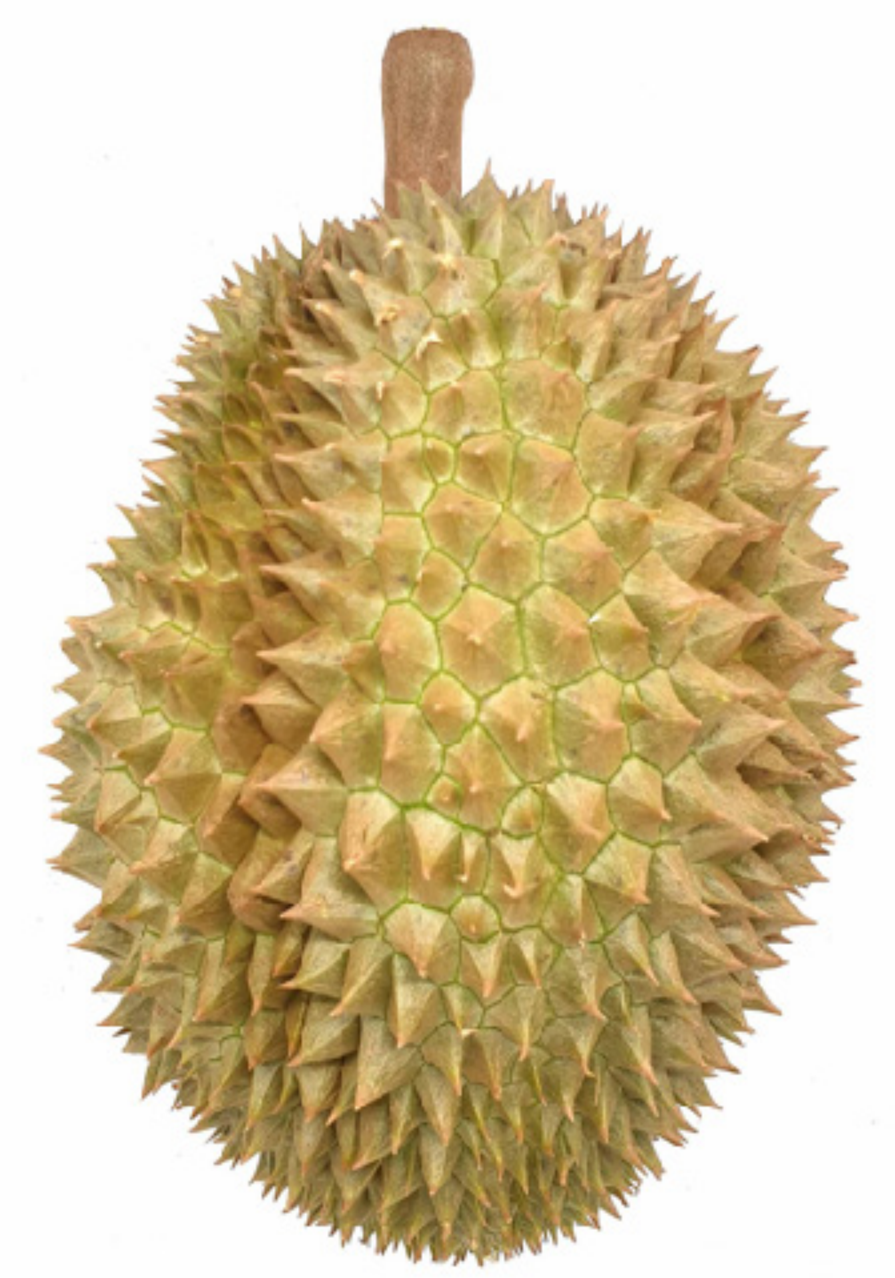


convergence

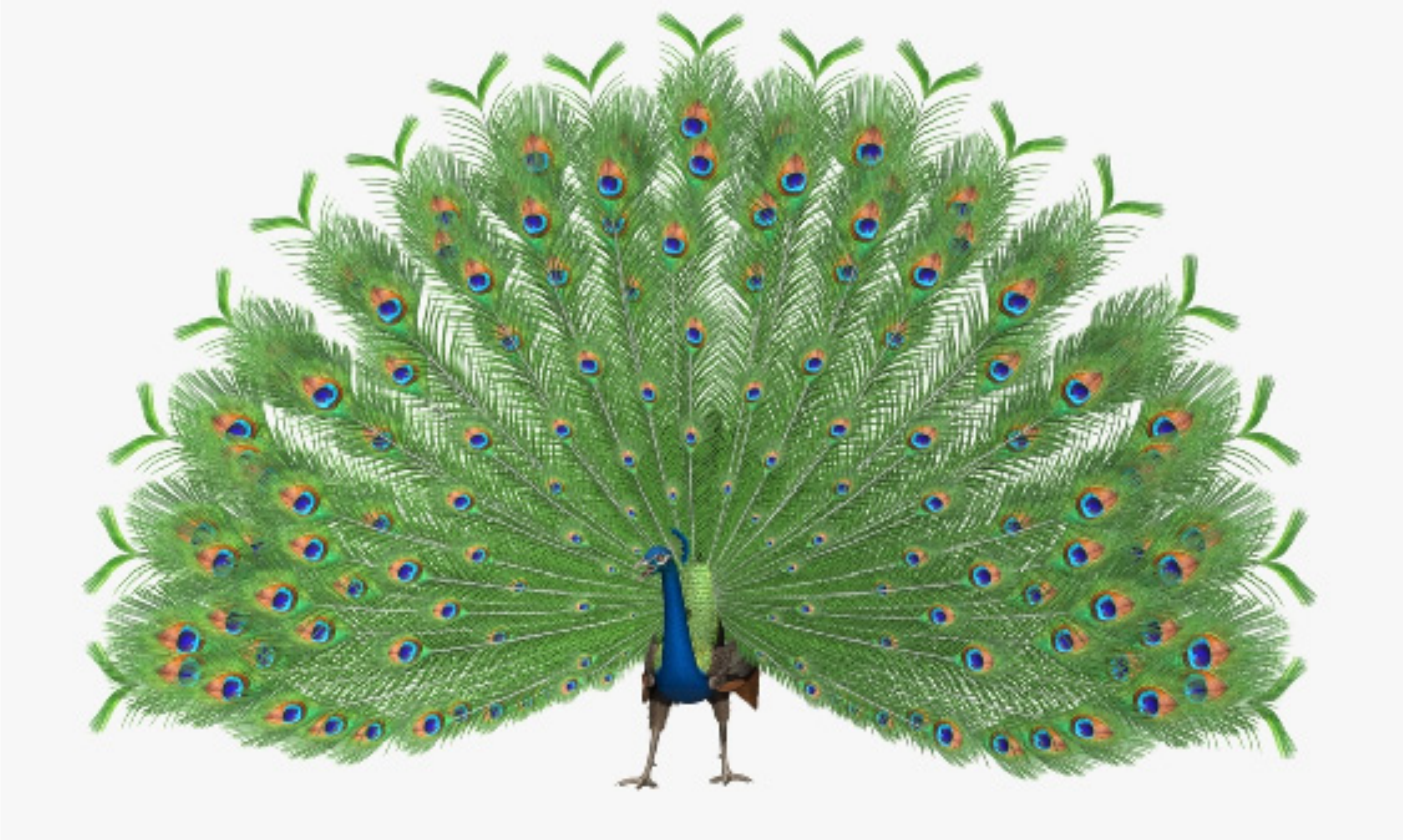
Deep Geometric Texture Synthesis



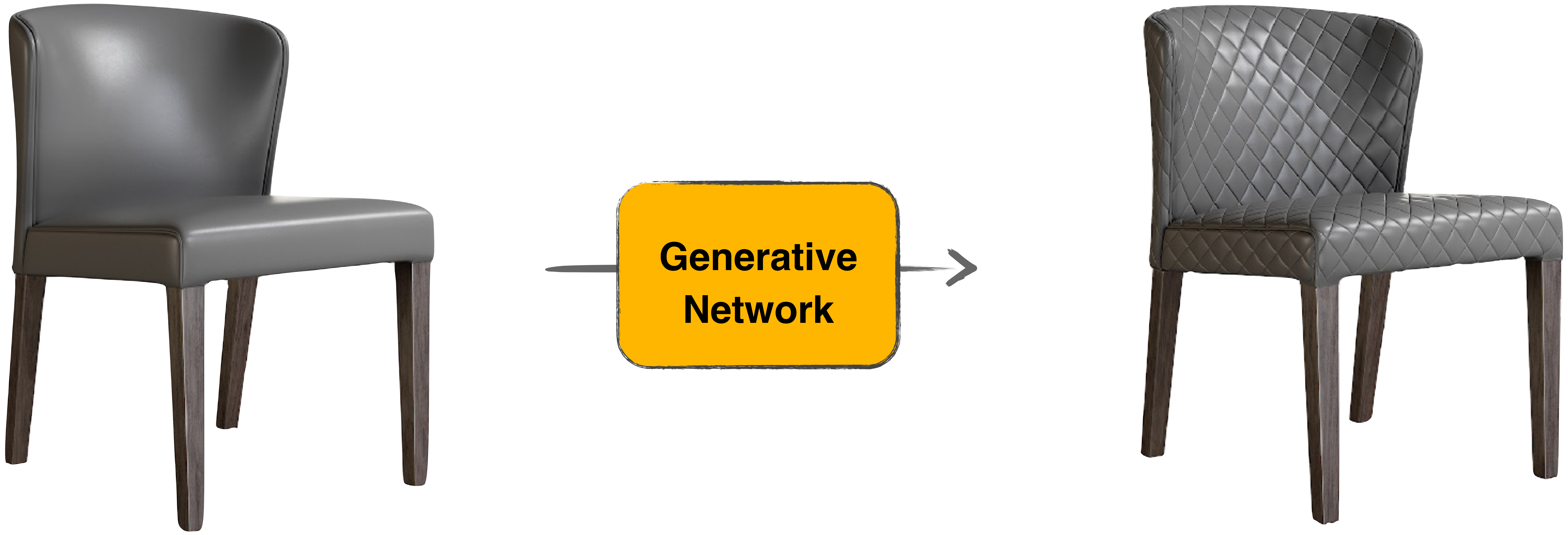
Geometric Textures



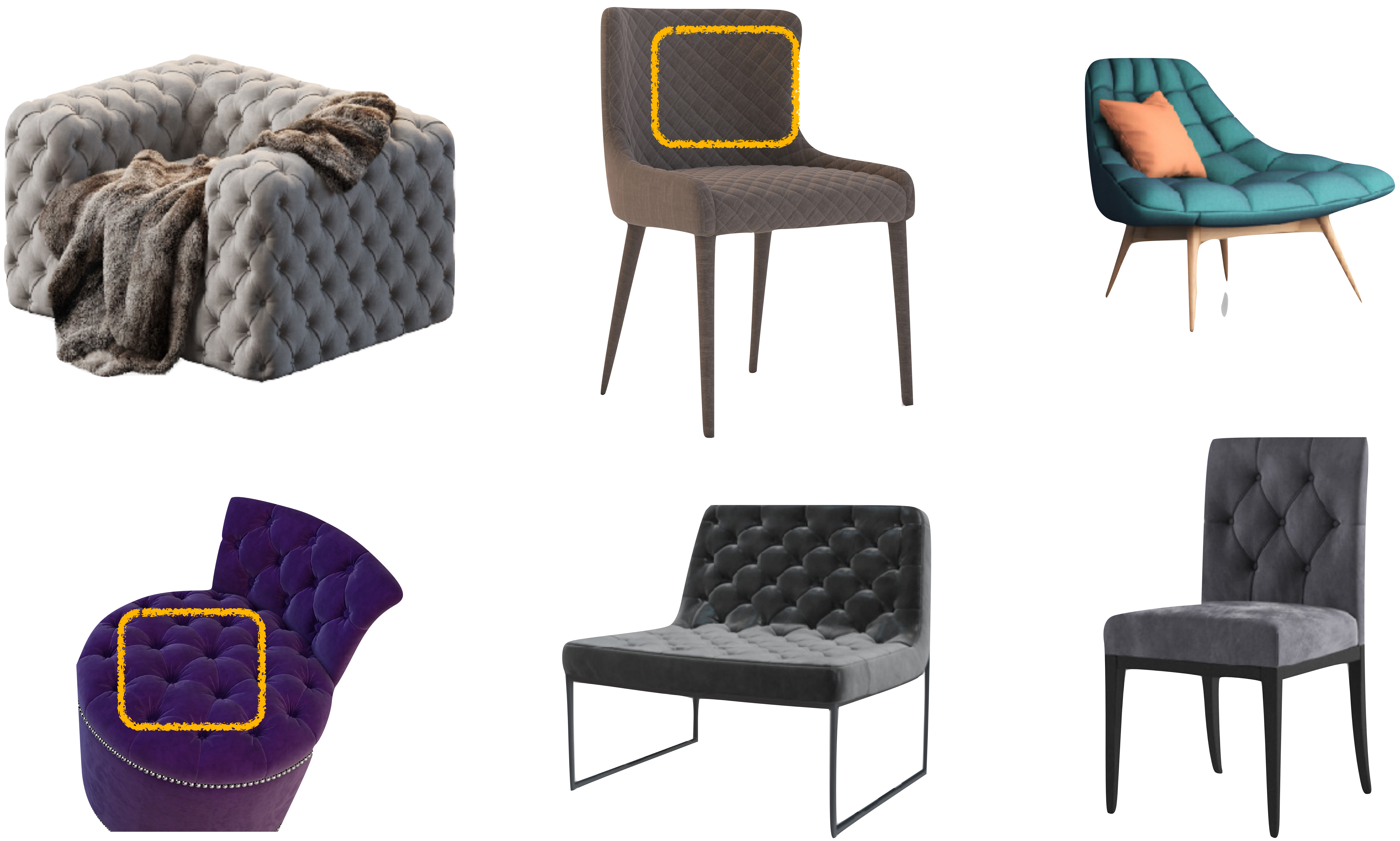
BIGHORN 2.0
MAXXIS



Objective: Generative Texture Model



External Dataset



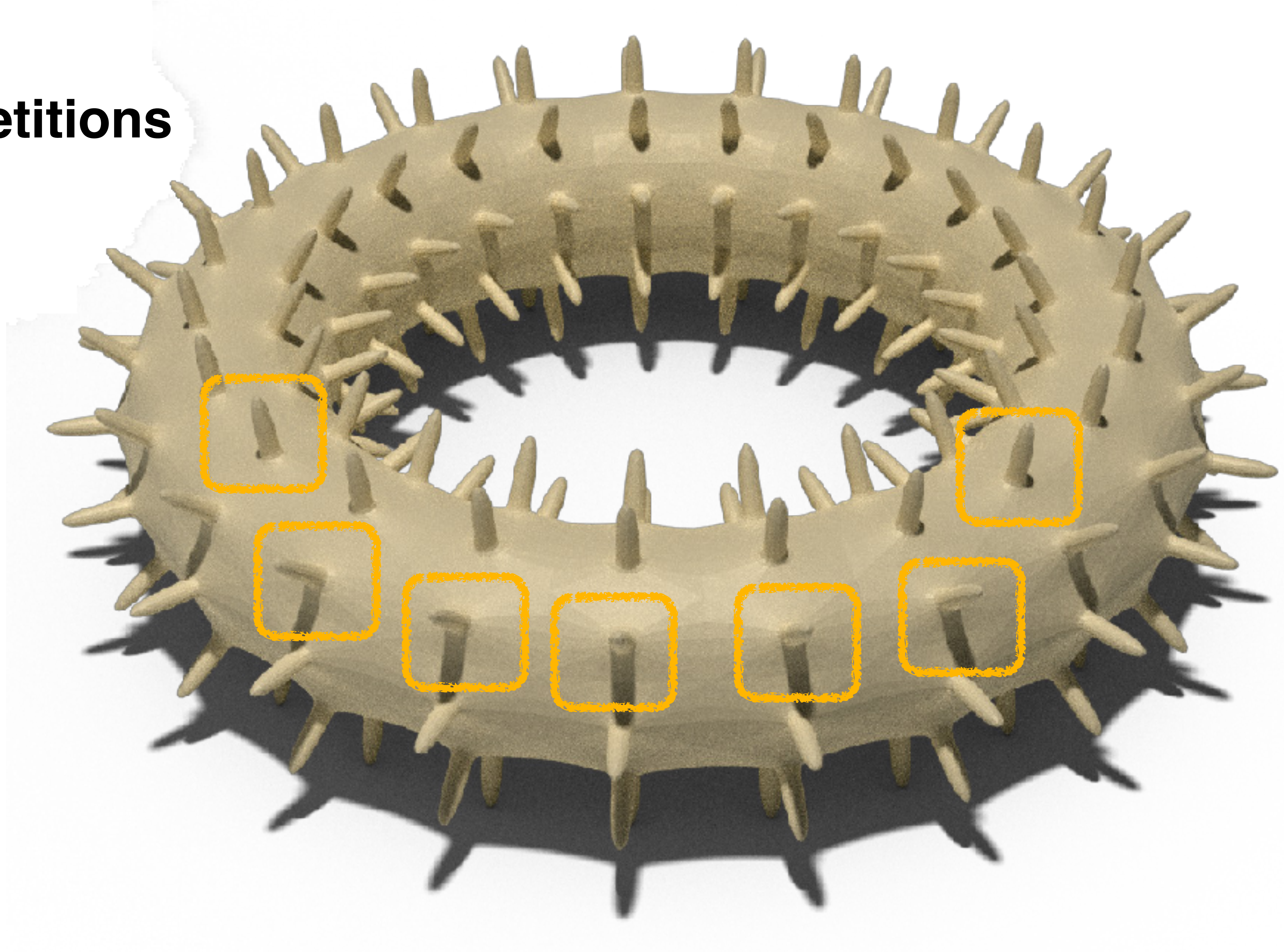
Difficult to obtain collections with same geometric textures

Inconsistencies make learning difficult

Learn from a Single Shape

Patches are training data

Textures contain self-repetitions

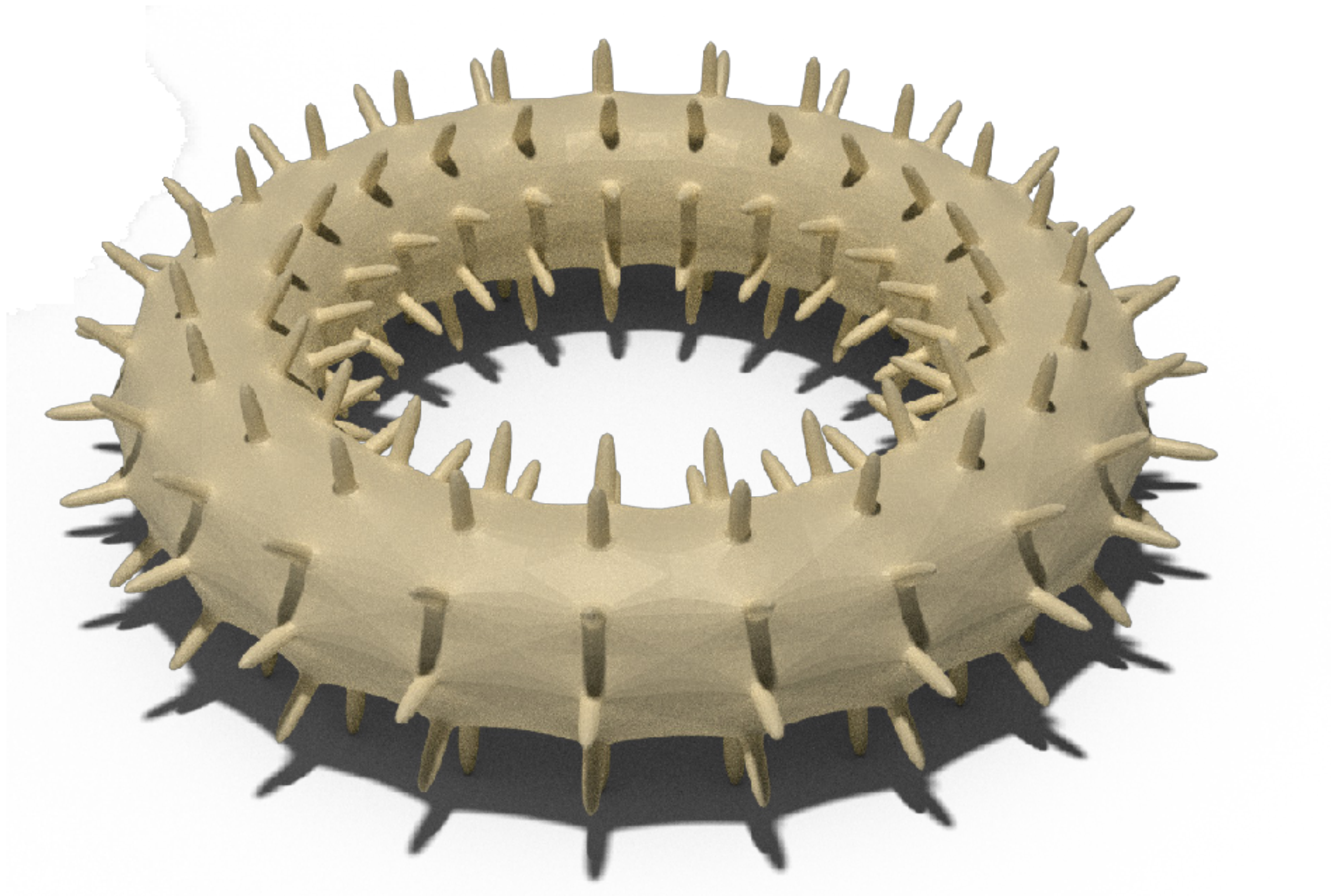


Learn from a Single Shape

Patches are training data

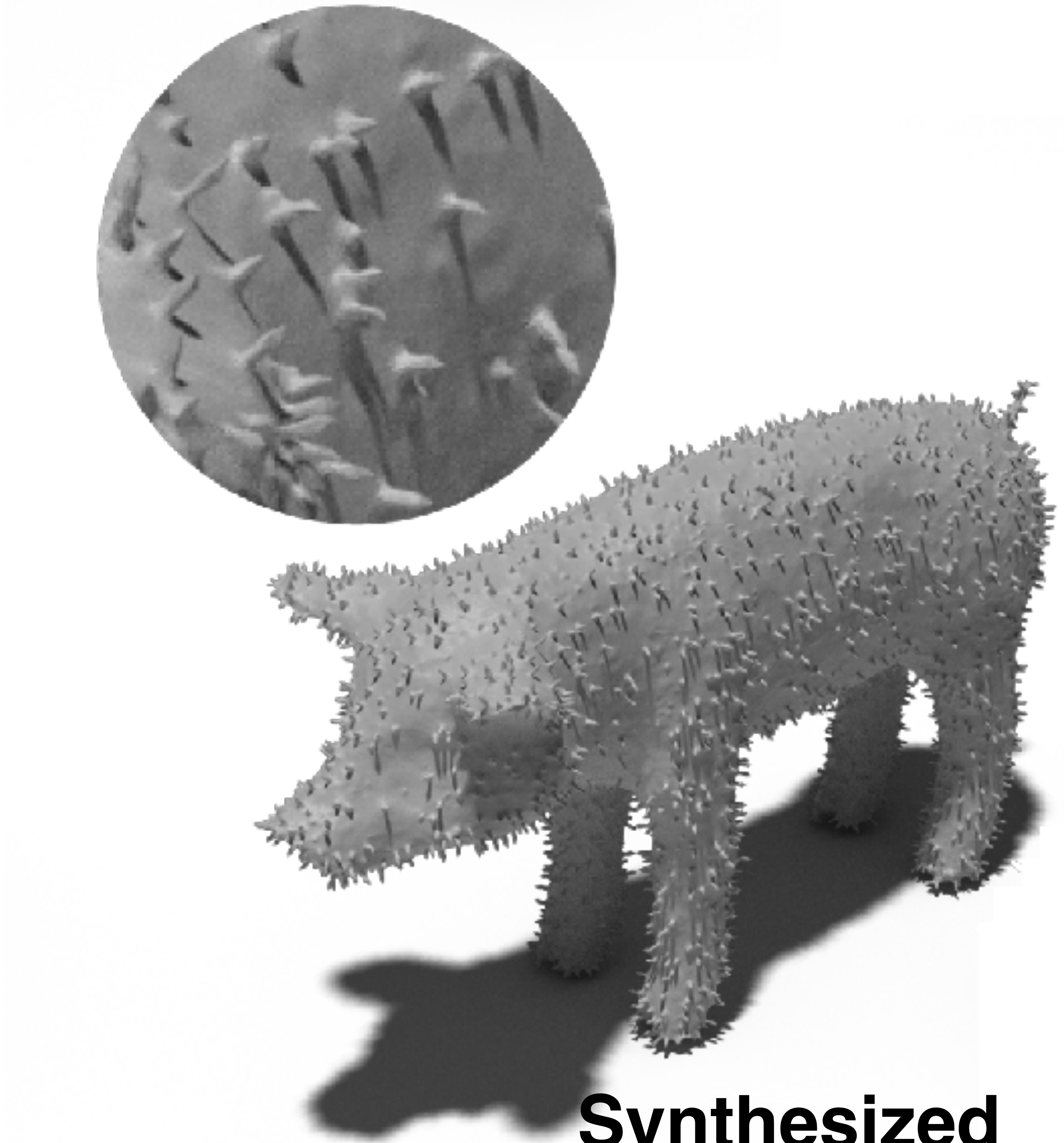
Textures contain self-repetitions

Local patches are similar for different shapes!



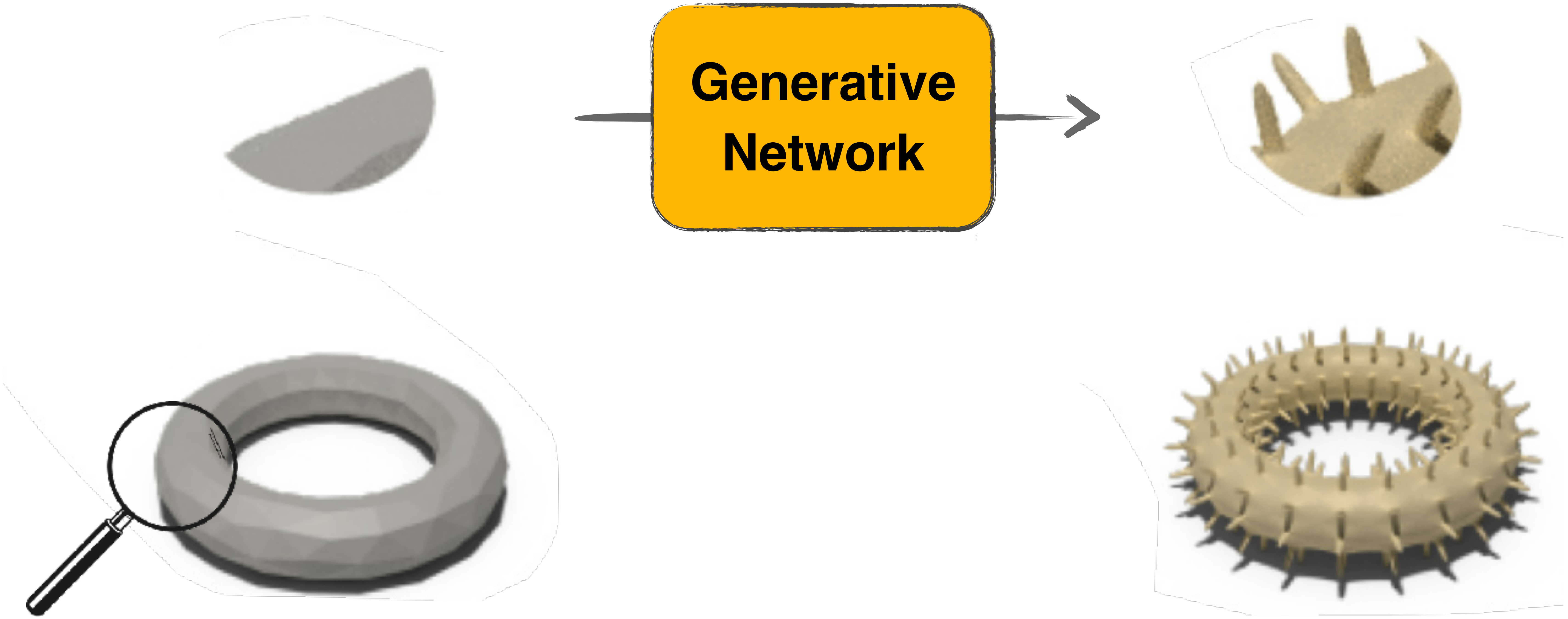
Exemplar

New Object



Synthesized Textures

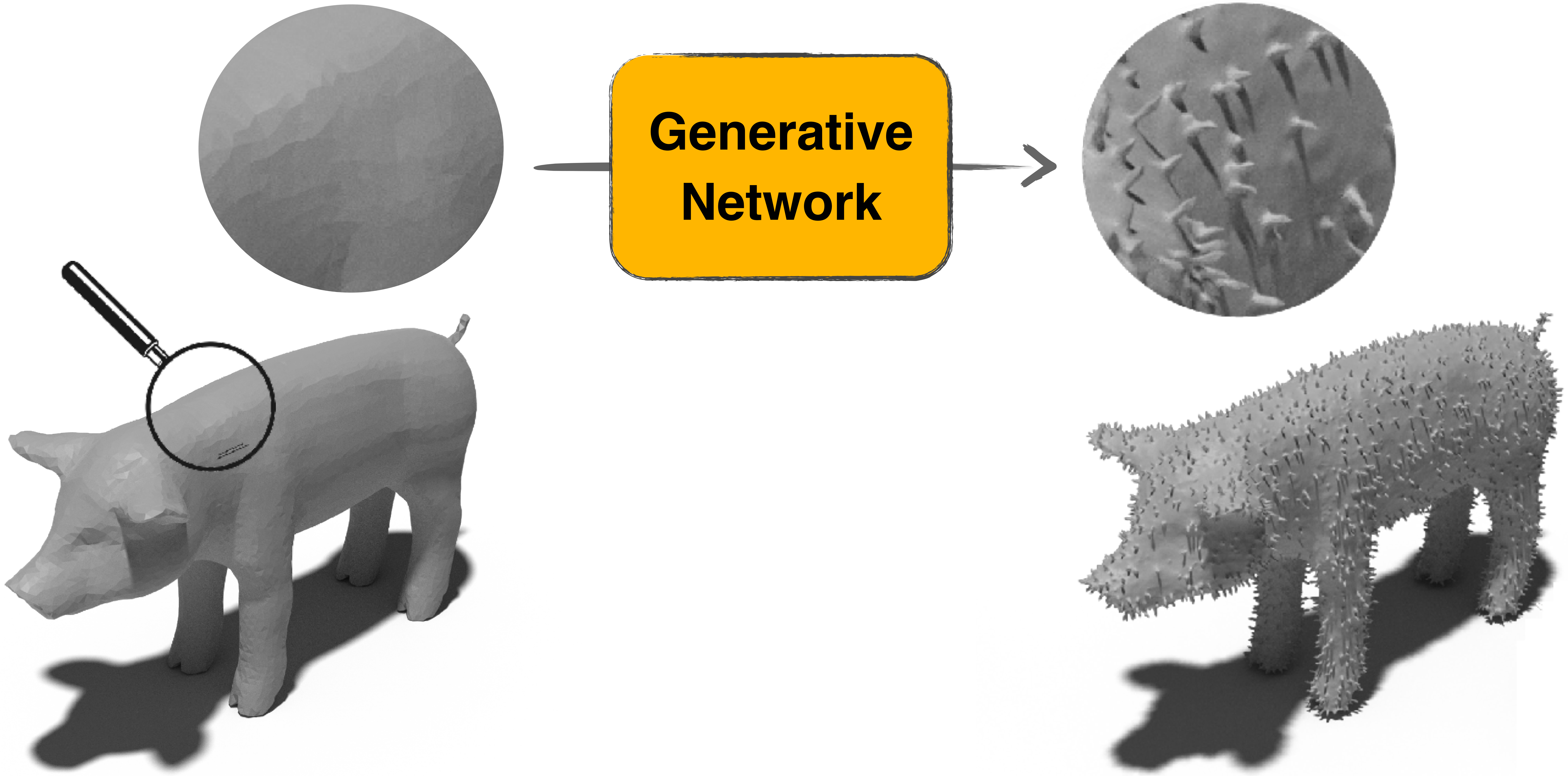
Train on local patches



Untextured

Exemplar

Generalization from local patches

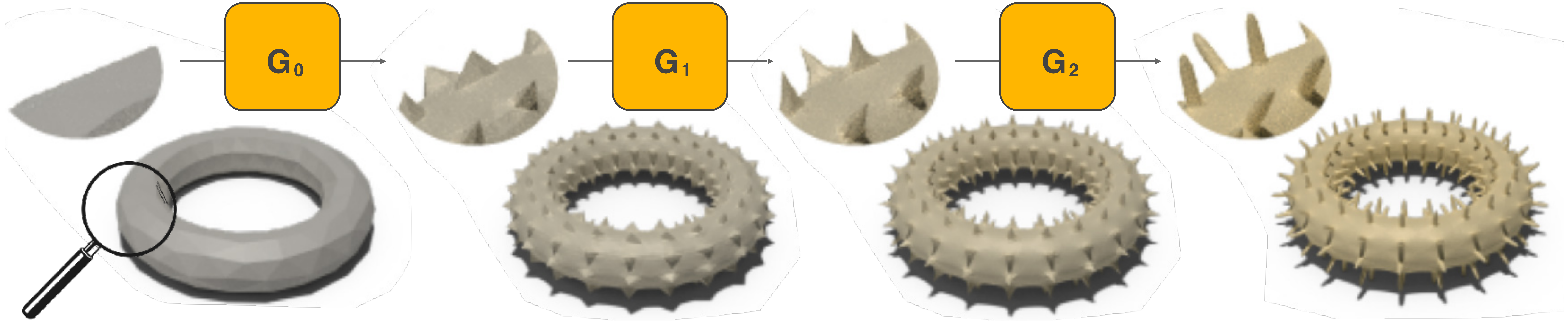


Generative Network

New Object

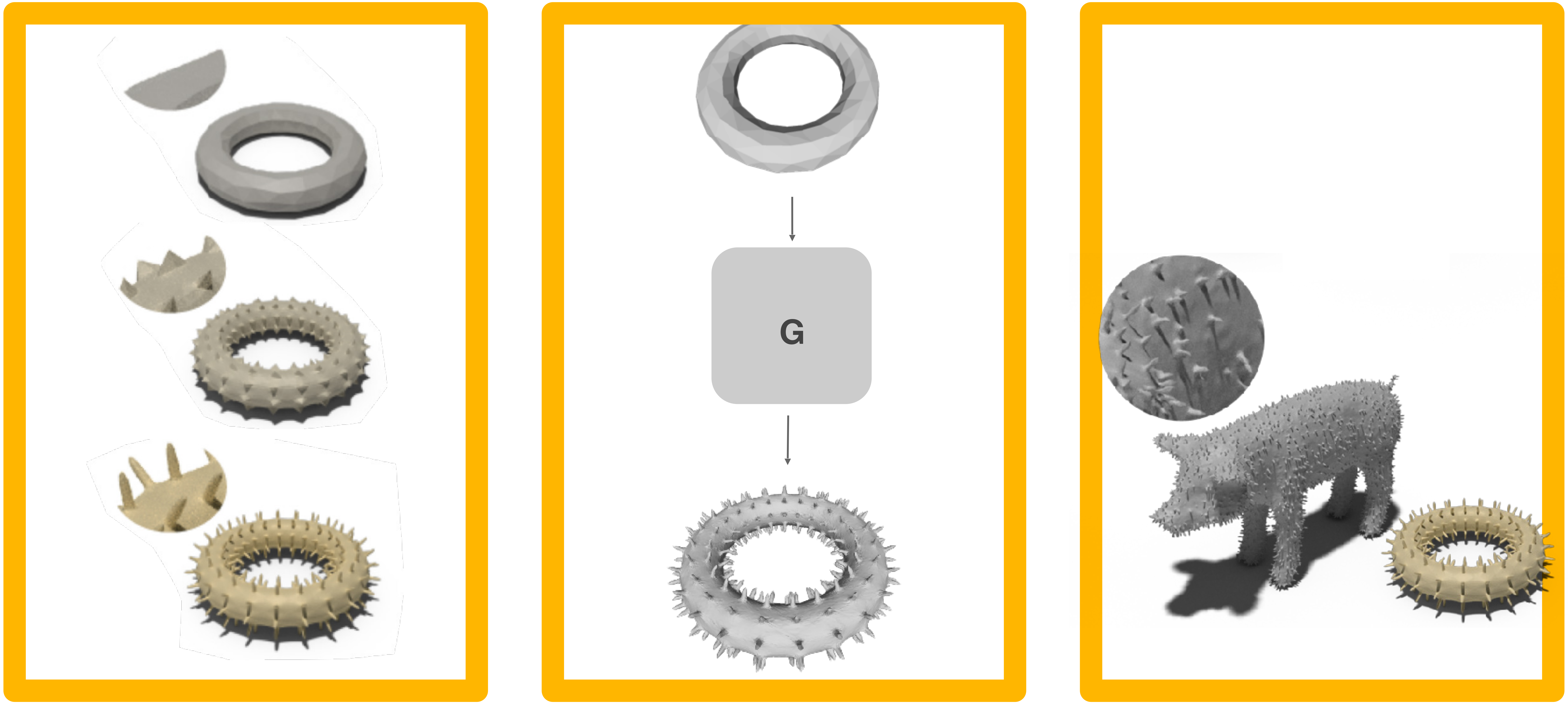
Synthesized

Progressive Texture Synthesis



Untextured

Exemplar



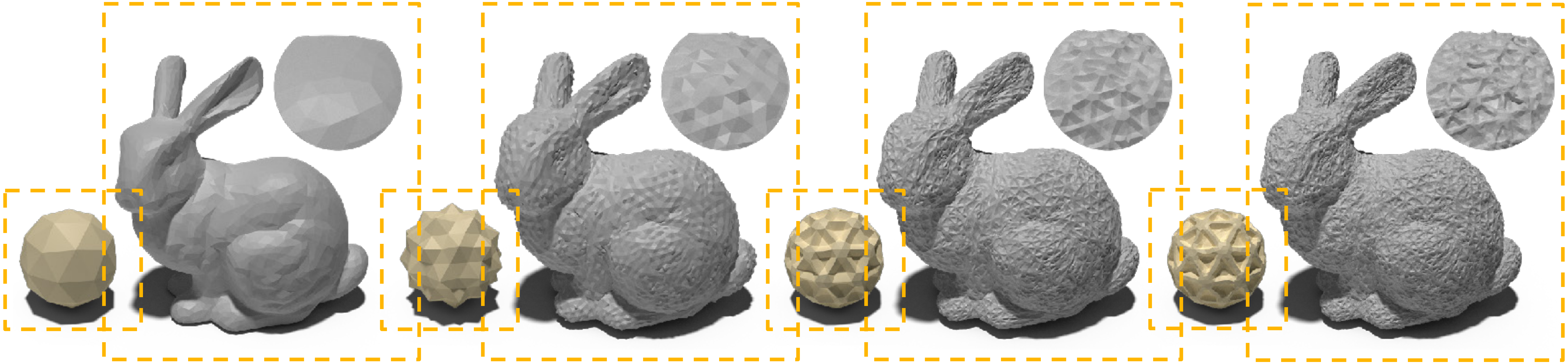
Multi-Scale Training Inputs

Progressive Training

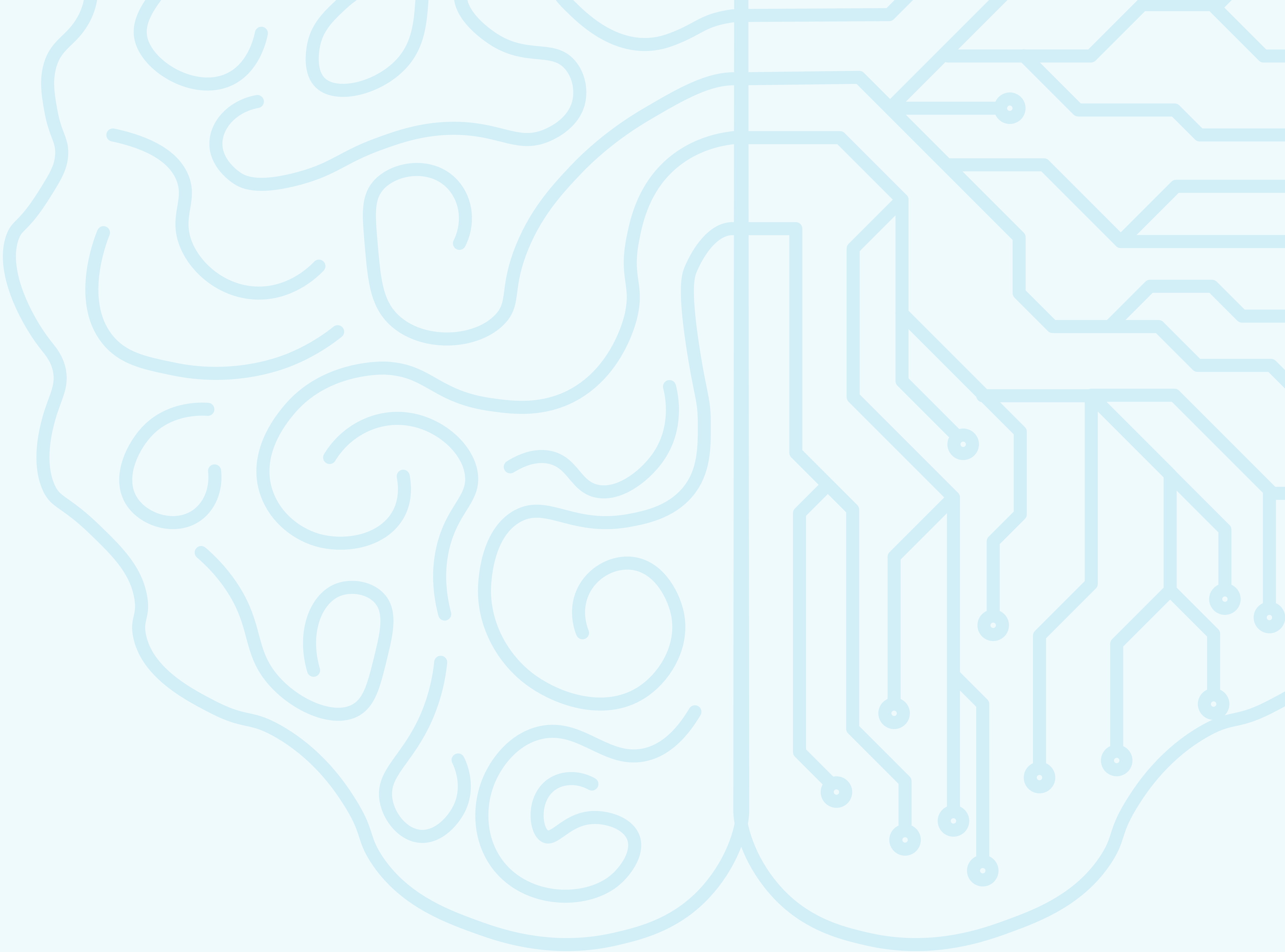
Inference

Progressive Texture Synthesis

Multi-Scale Textures

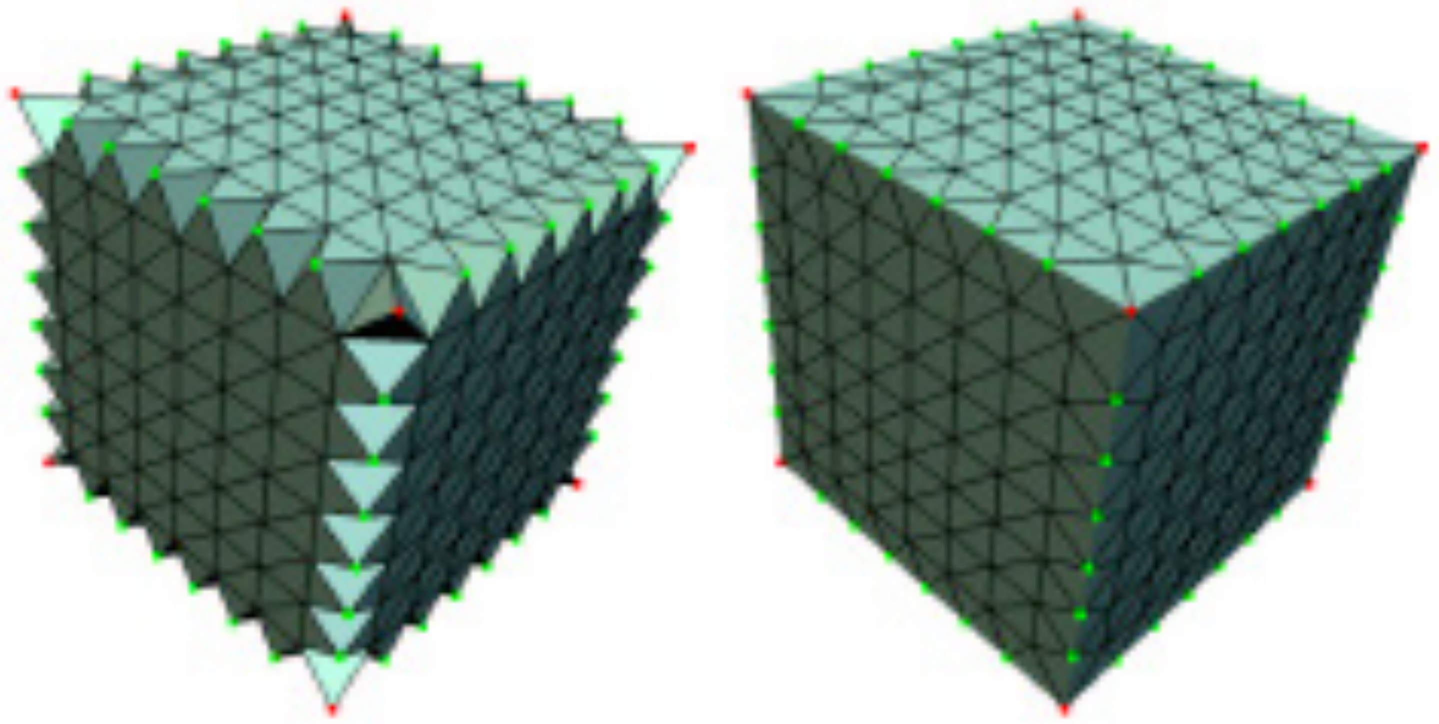


Texture Interpolation

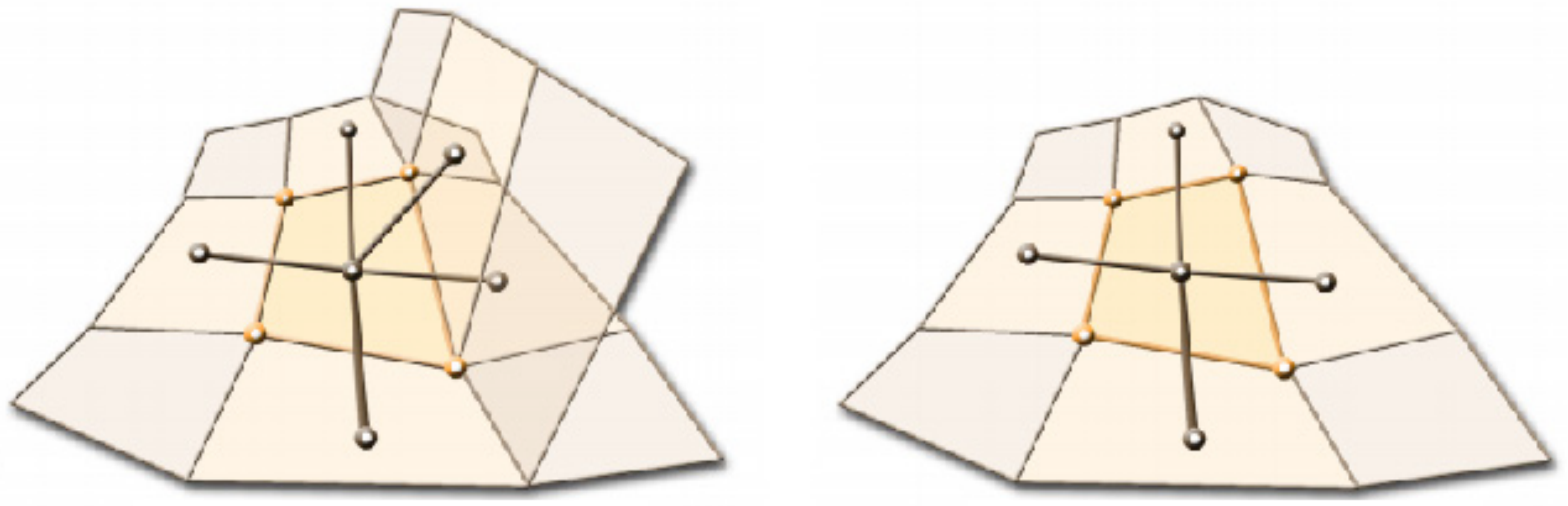


Summary

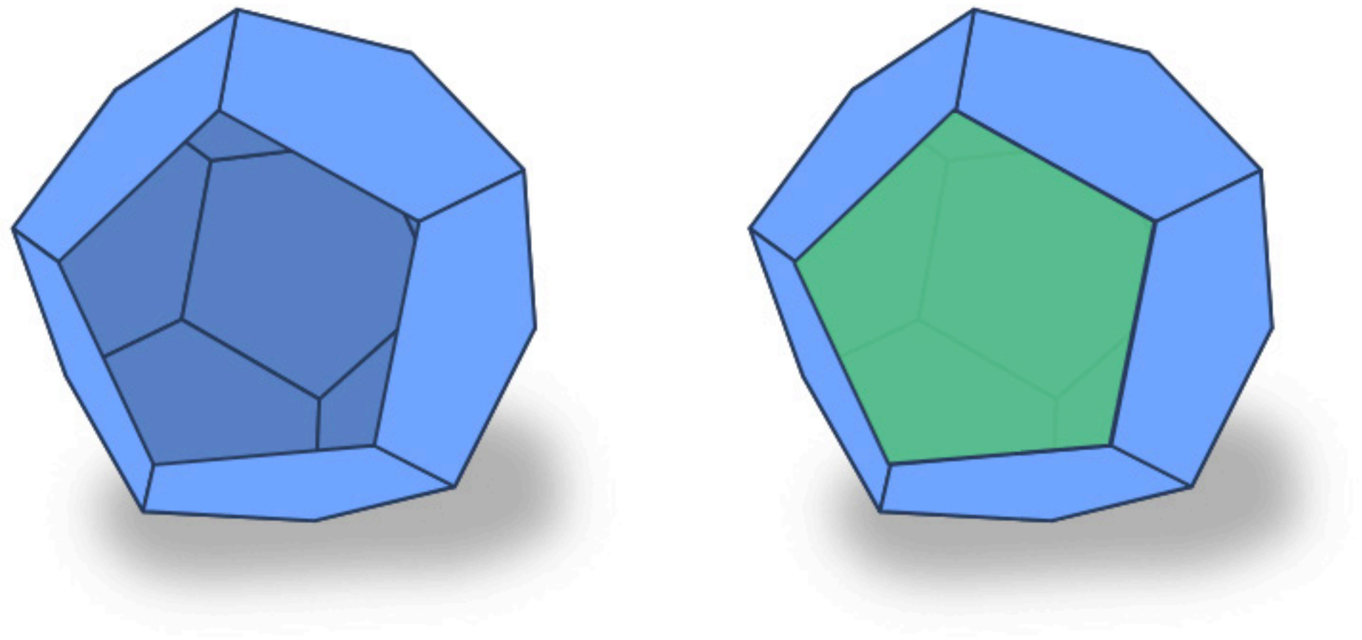
Connectivity based Mesh Learning



sharp features

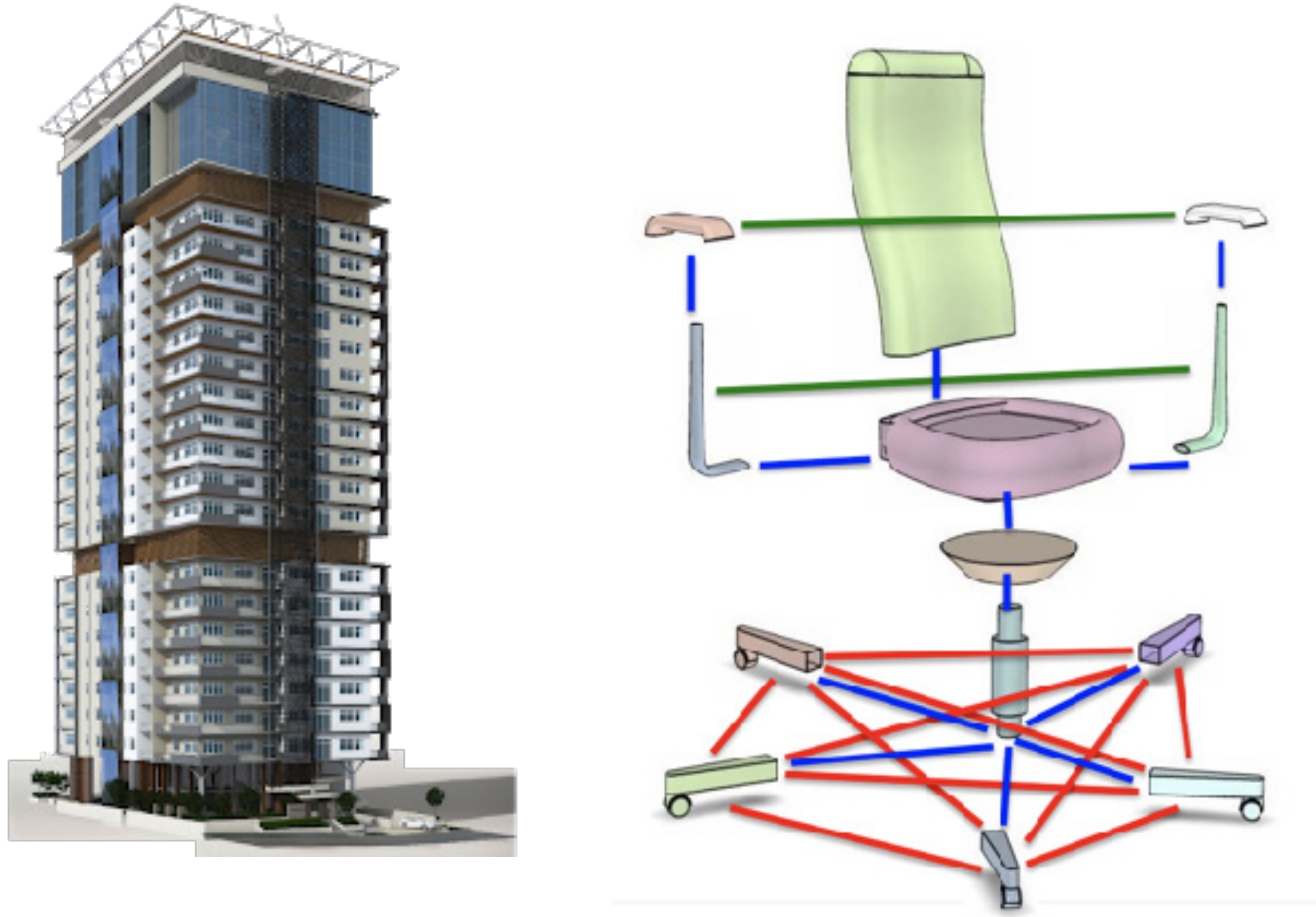


fix non-manifold

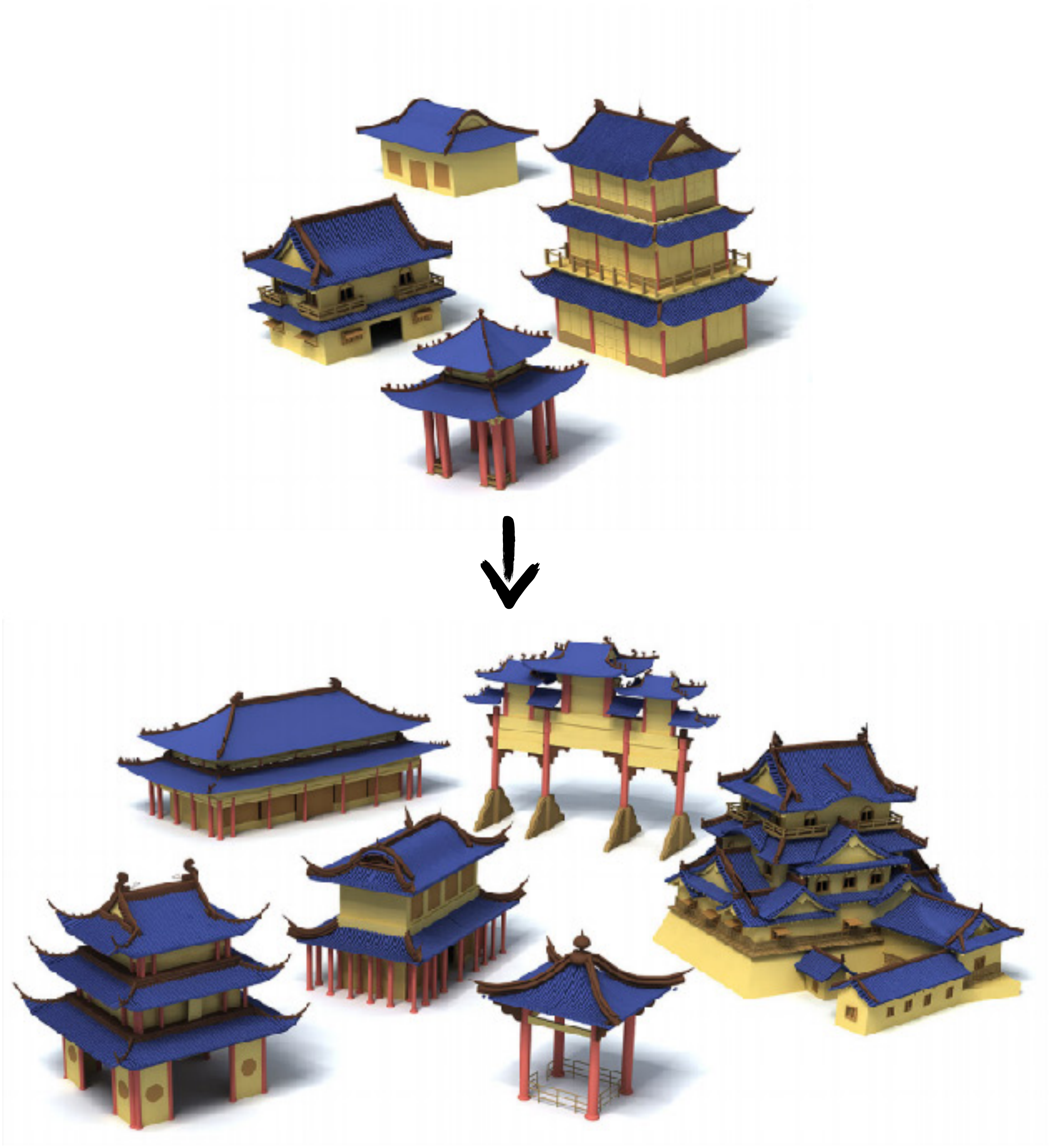


hole filling

Mesh Generative Models



structural learning



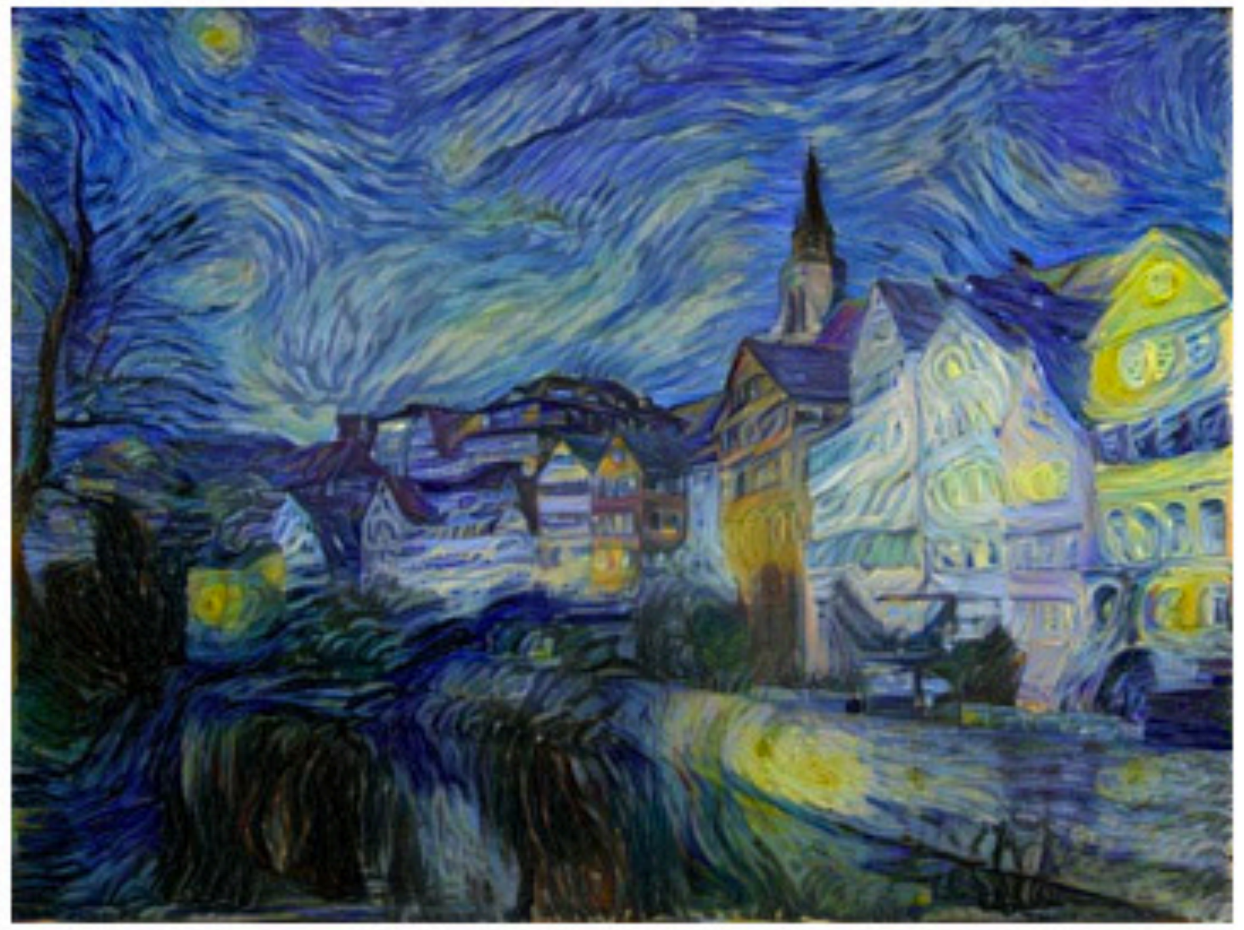
structural shape synthesis

mesh boolean

Shape Perception

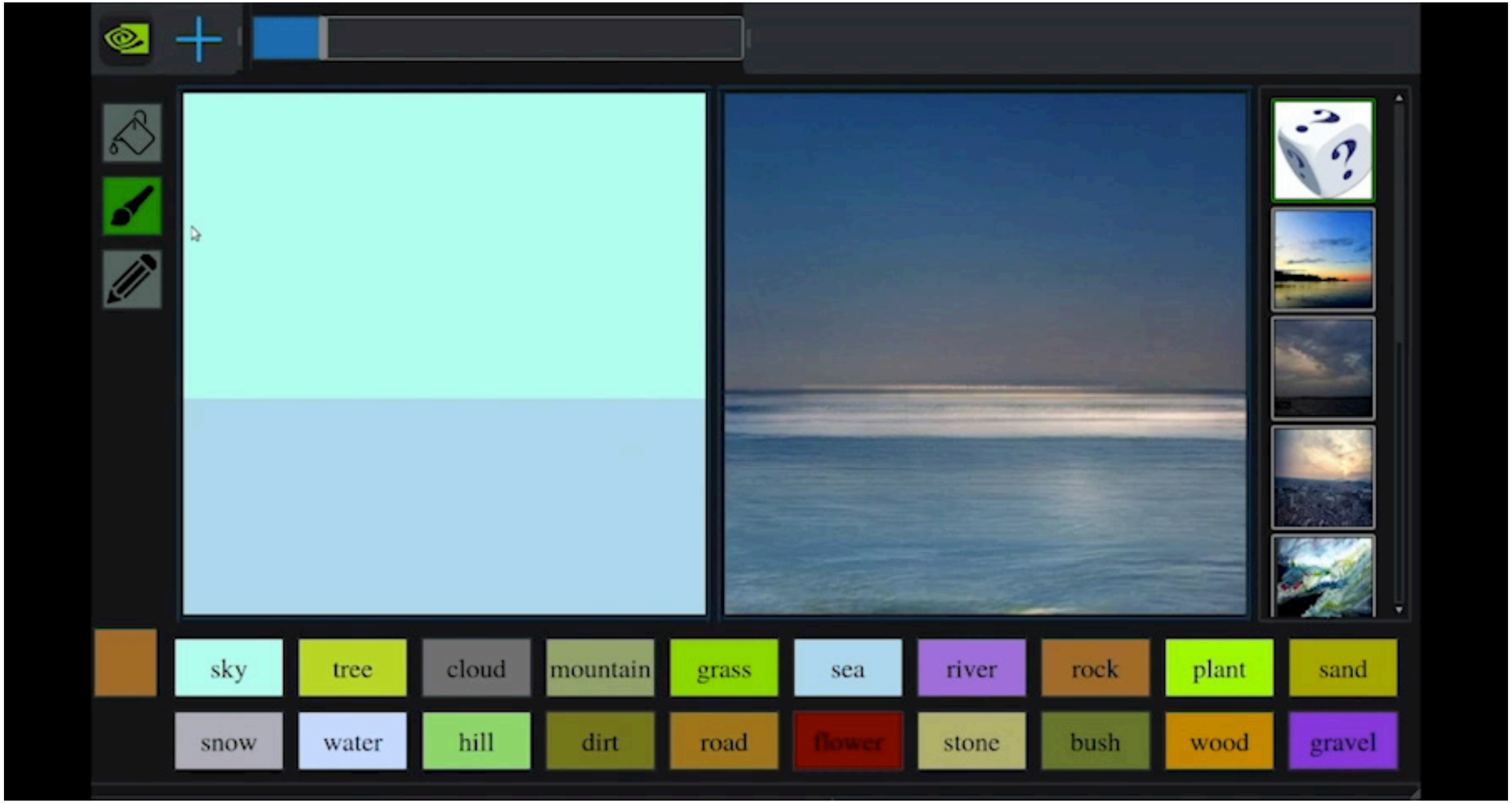
input

style



output

Futuristic 3D Modeling Tools



Park et al. 2019

Draw Inspiration from Classic Methods