Neural Deformation, Parameterization and Compression of Polygonal Meshes

Vladimir (Vova) Kim

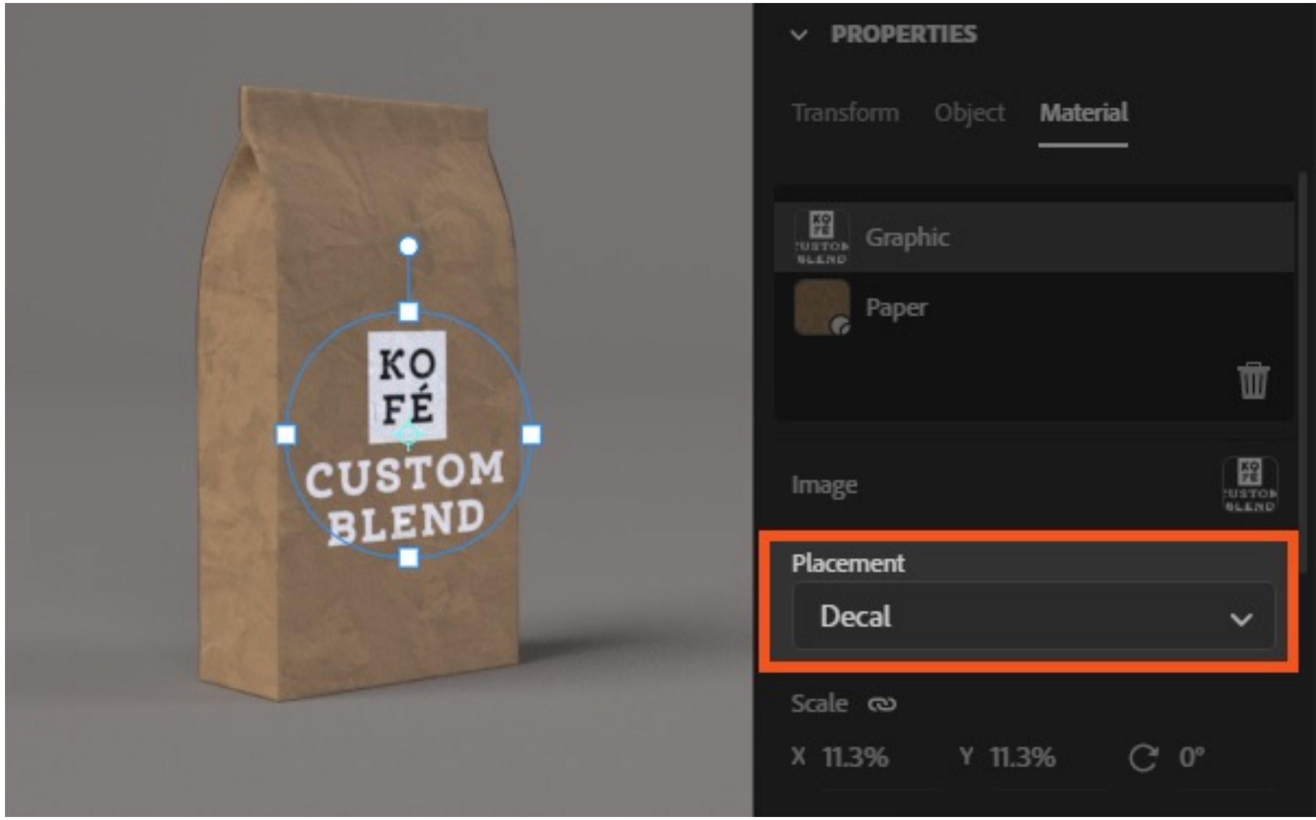
Adobe Research, Seattle

Adobe

Motivation

Content Creation

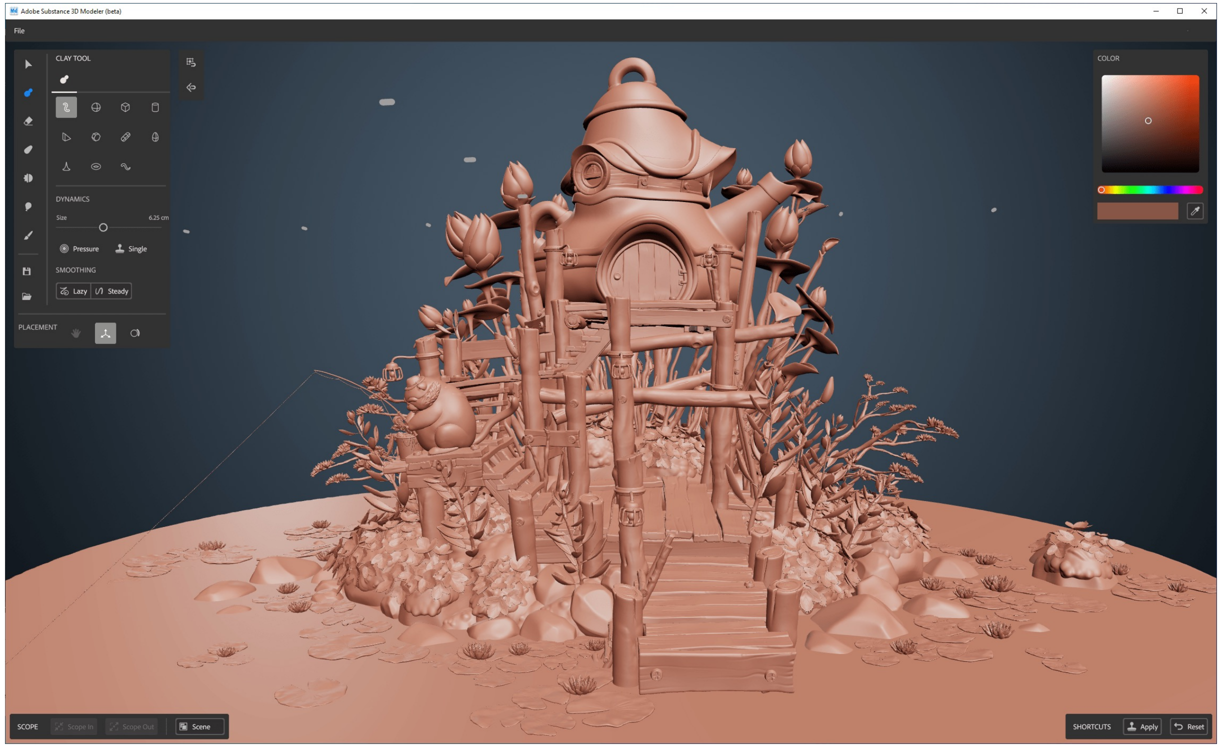
Adobe Substance Painter



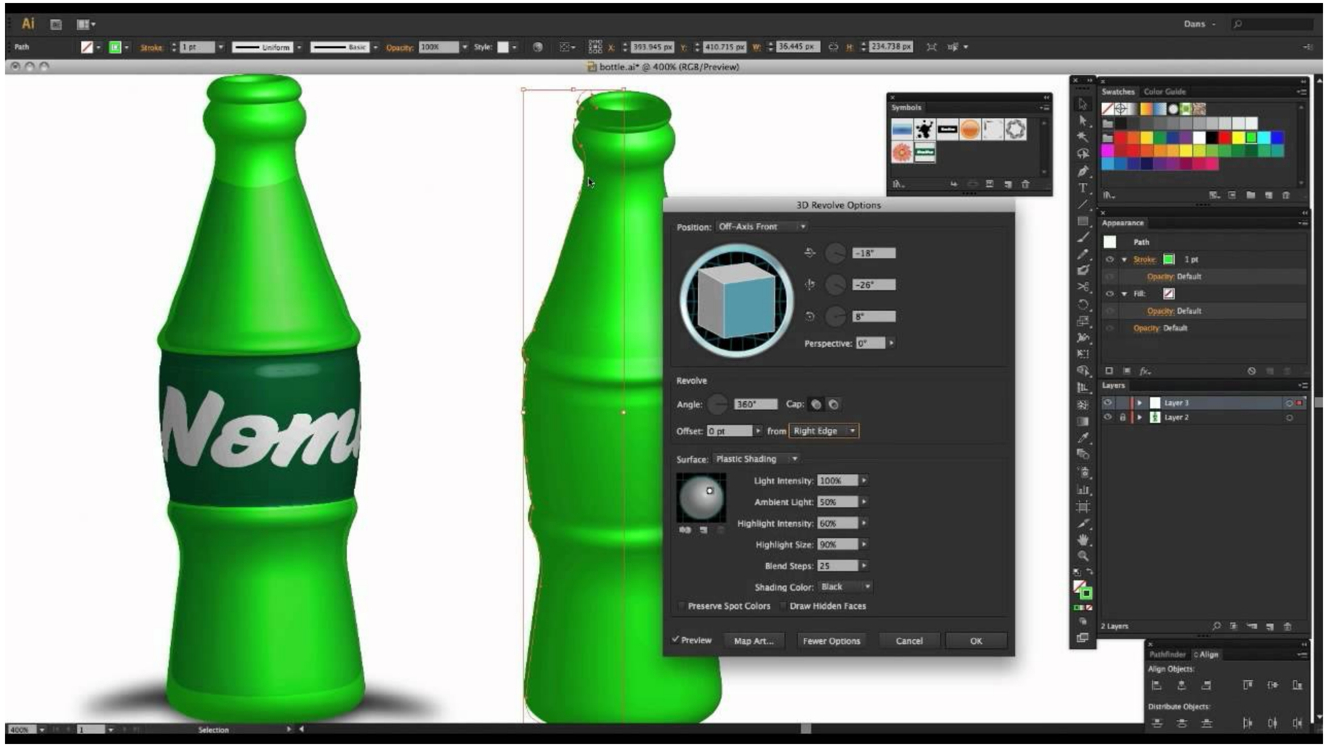
Adobe Stager

Motivation

Content Creation



Adobe Substance Modeler



Adobe Illustrator 3D

Why Polygonal Meshes?

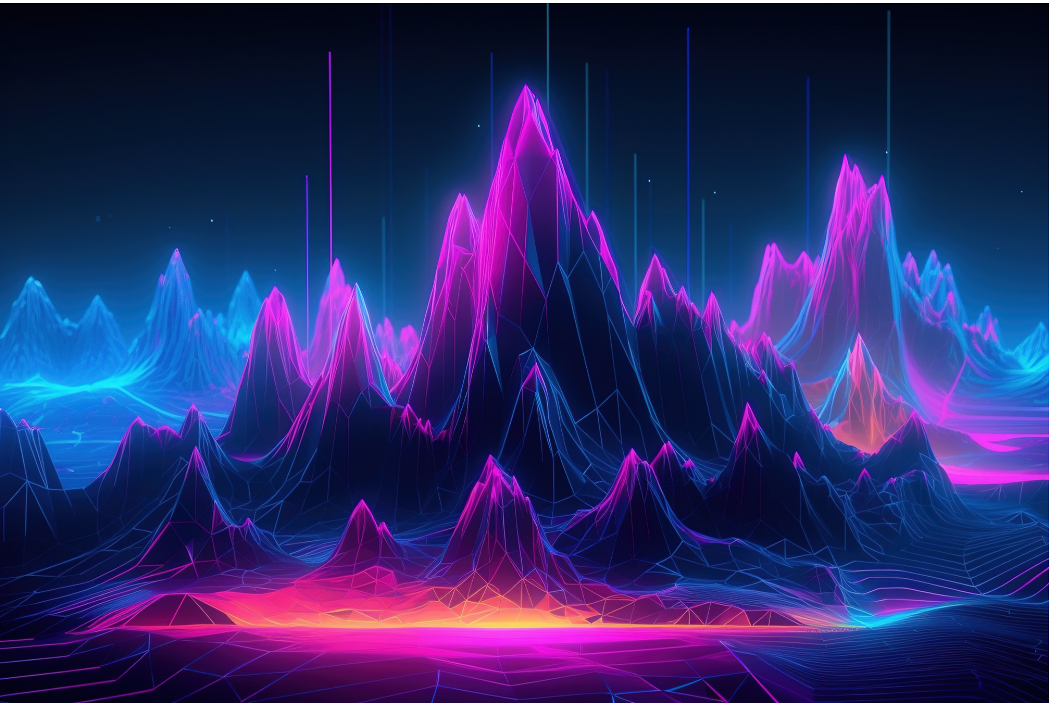
Concise (sparse) representation

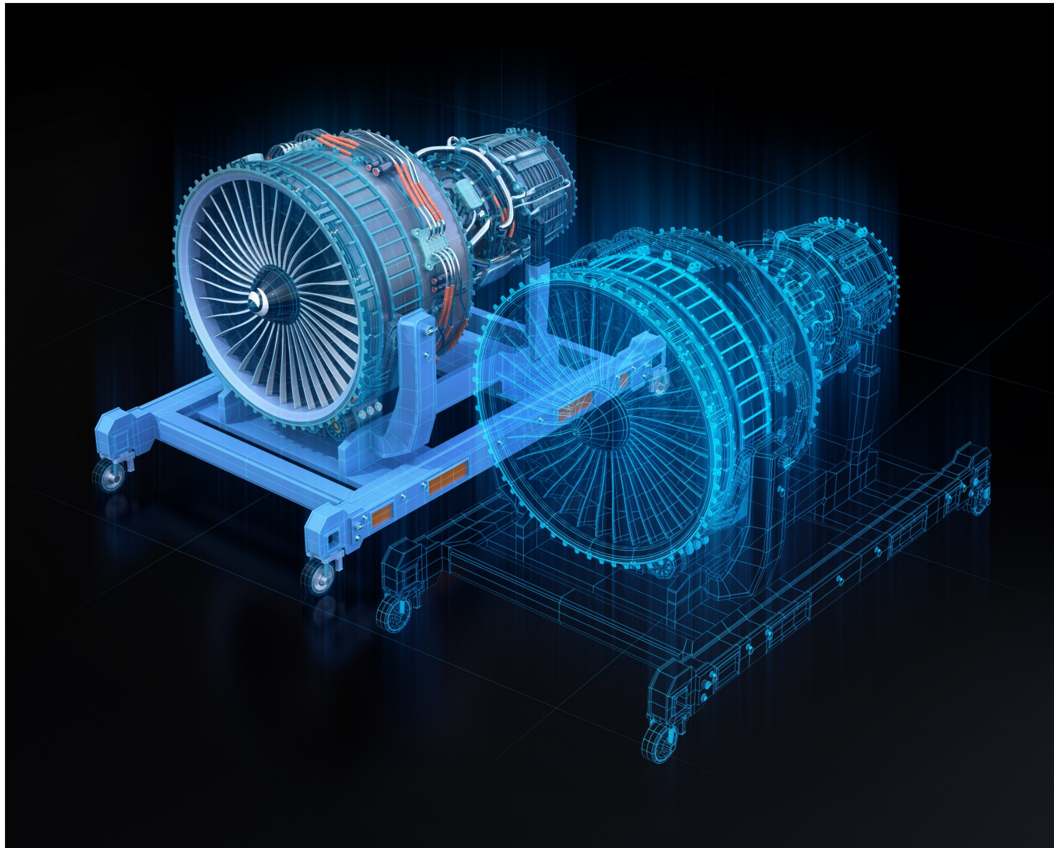
Factorized into materials and geometry

Concisely store spatially-variant materials (if parameterized)

Lots of available data

Supported by most existing workflows, pipelines, tools





Why Neural Networks?

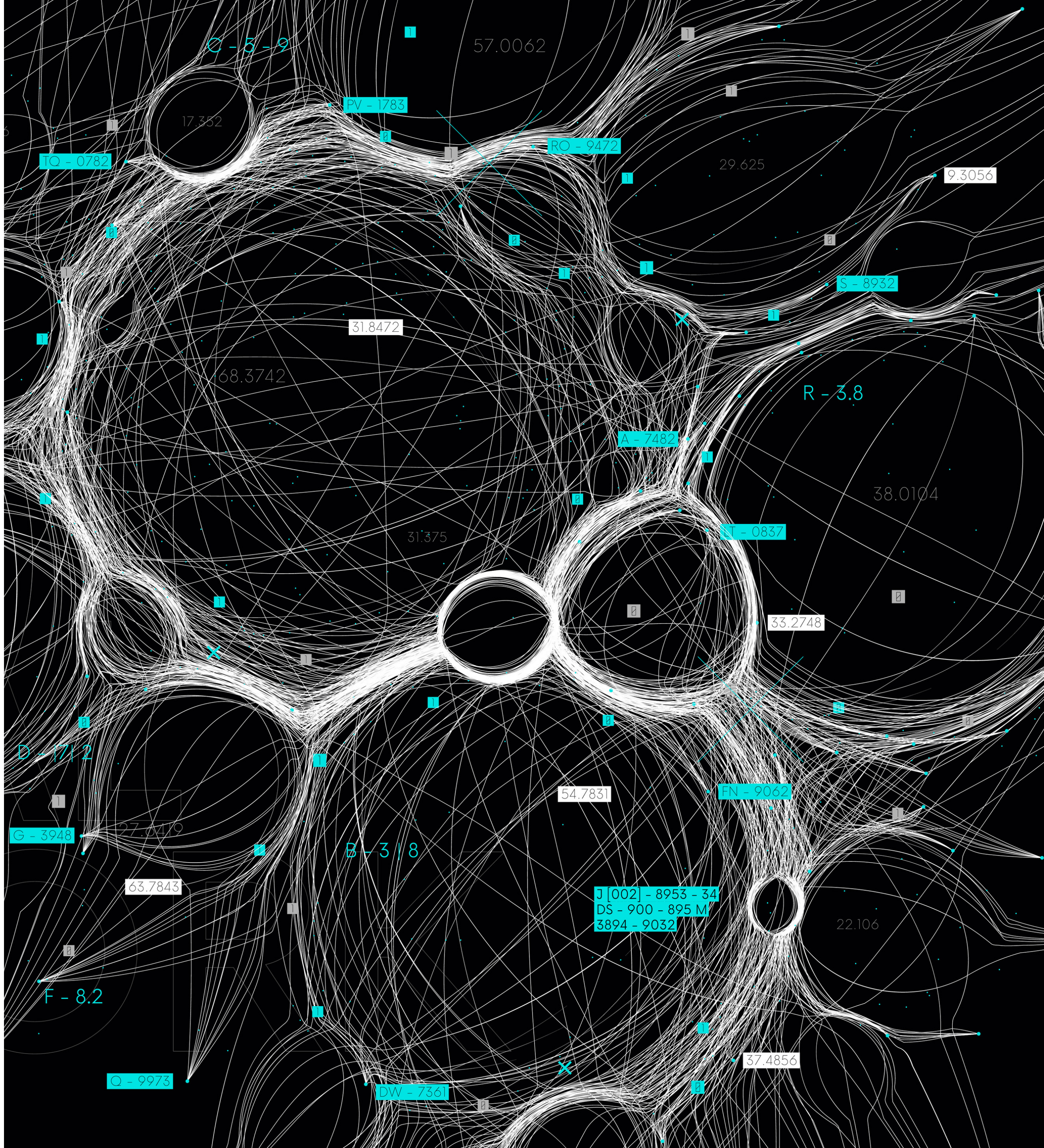
Encode complex priors

- Priors derived from human understanding
- Priors on how to optimize things better

Fully differentiable pipelines to prototype

- Variables to optimize
- Objective functions
- Representations

Universal toolbox to share with others



Neural Deformation

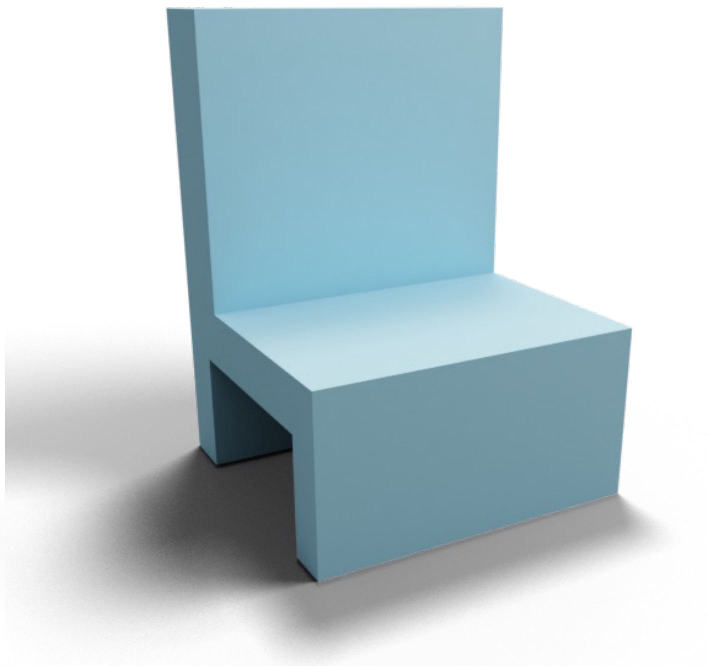
- Deform the source to match the target while preserving the details

Neural Deformation

- Naïve approach:

$$f_\theta : z_{\text{source}} \times z_{\text{target}} \times \mathbb{R}^3 \to \mathbb{R}^3$$

Source



Target

Groueix et al. CGF 2019

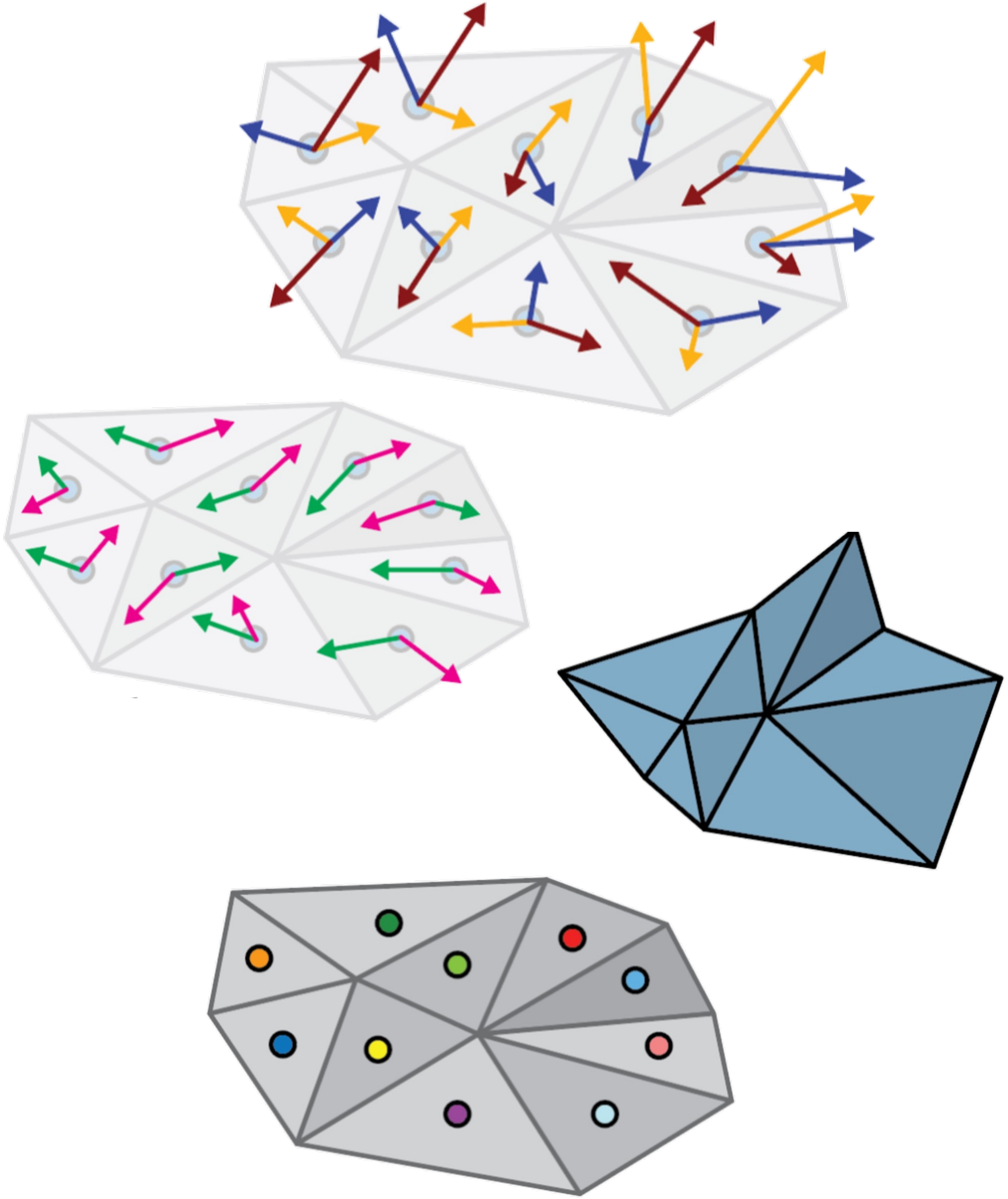
Why Geometry Processing?

Encode simple priors and constraints

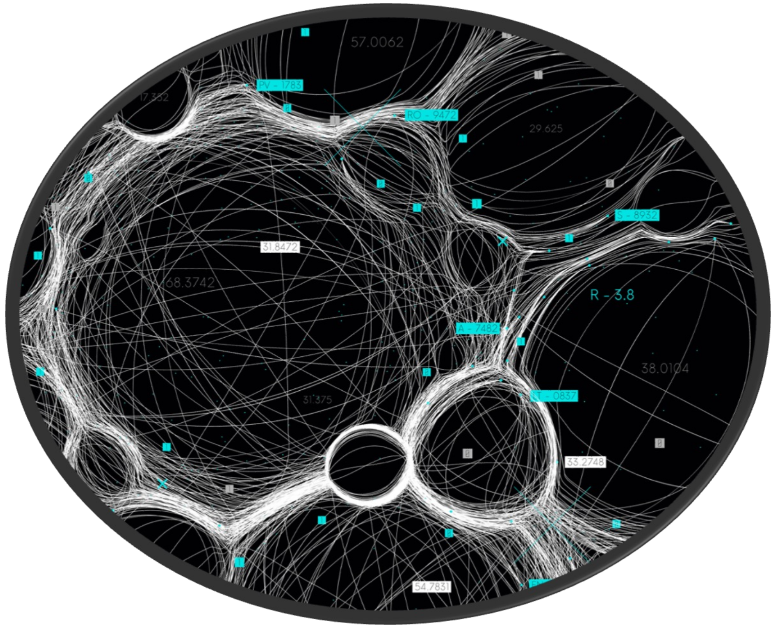
Mature mathematical foundations

Operators defined on irregular domains

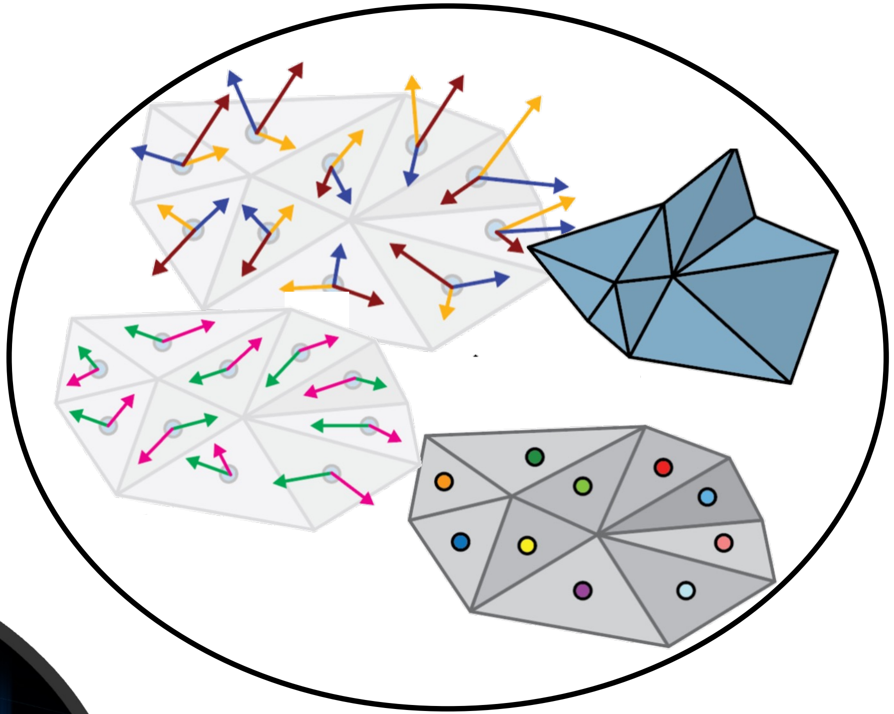
Often offer simple reusable tools



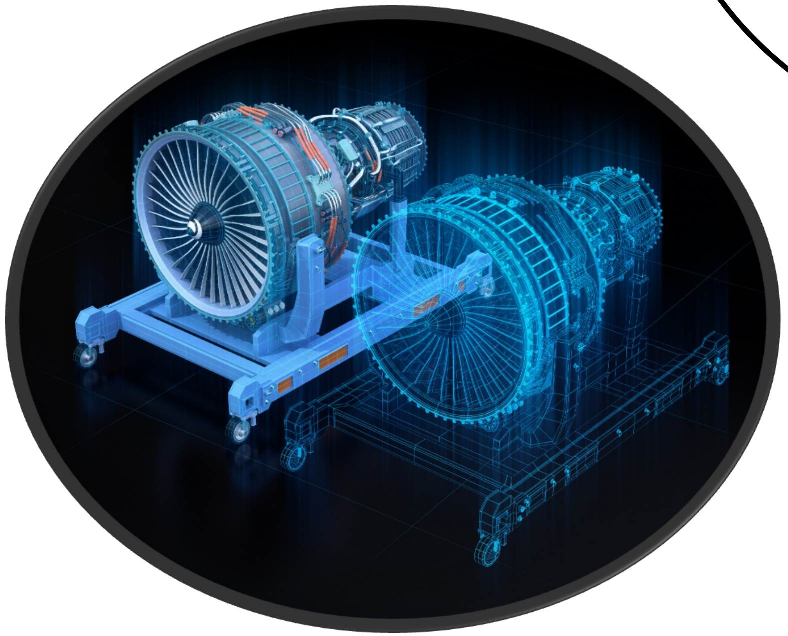
Powerful Combination



Neural Networks

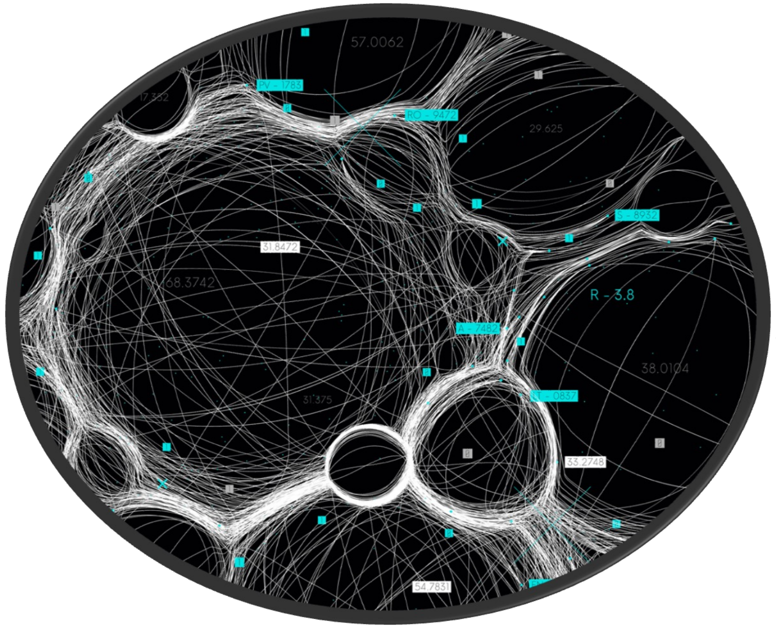


Geometry Processing

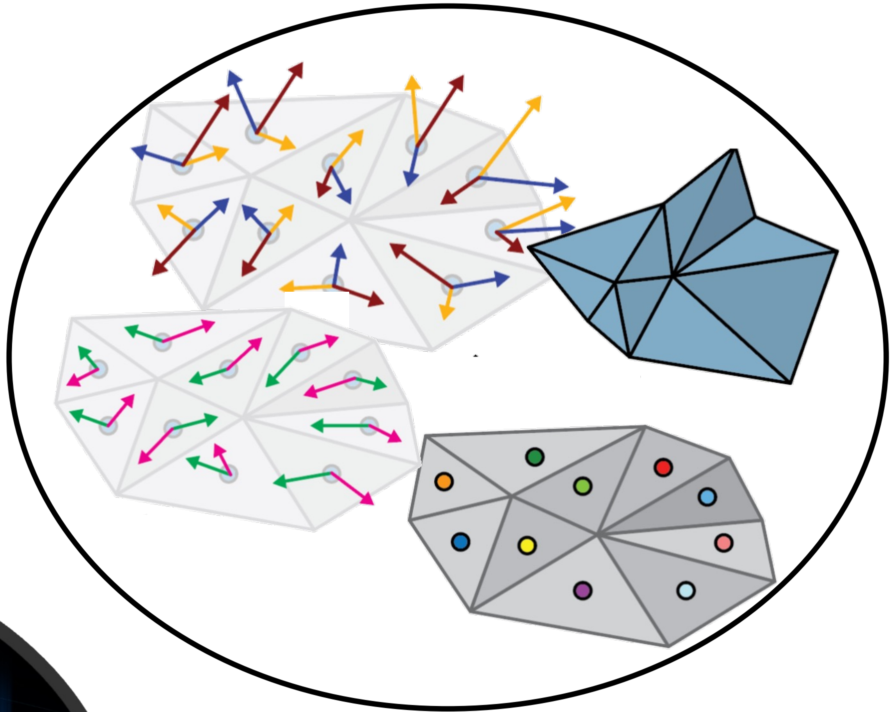


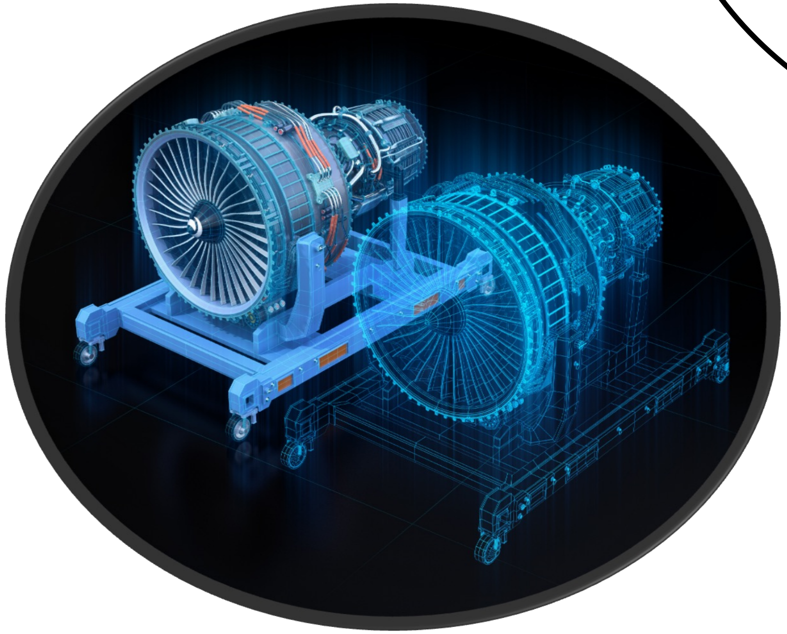
Meshes

Geometry Processing Helping Machine Learning



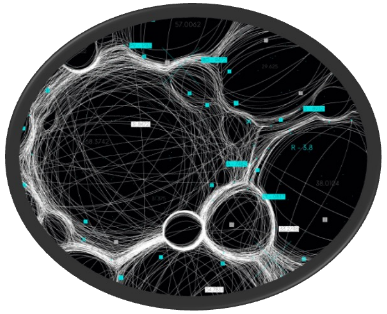
Neural Networks


Geometry Processing

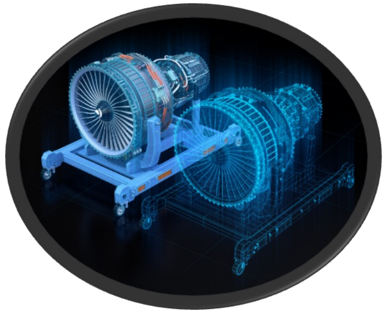

Meshes

Neural Deformation

- Naïve approach:

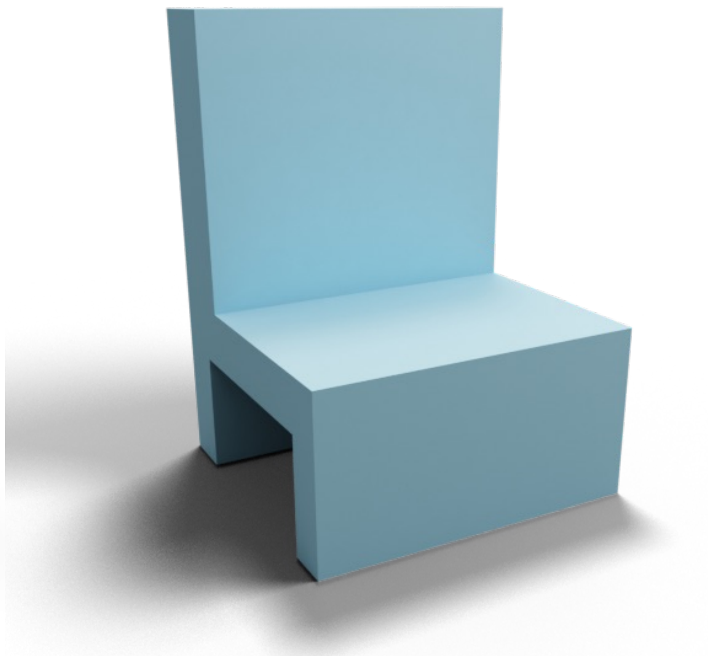


+



$$f_\theta : z_{\text{source}} \times z_{\text{target}} \times \mathbb{R}^3 \rightarrow \mathbb{R}^3$$

Source

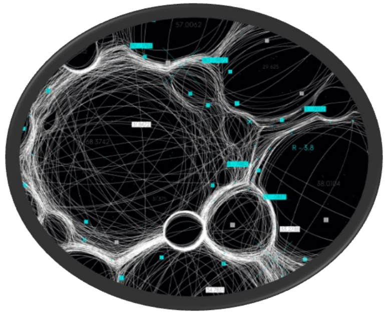


Target

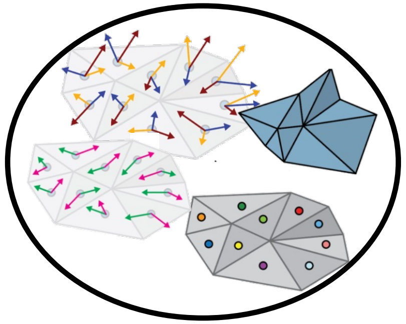
Groueix et al. CGF 2019

Neural Deformation

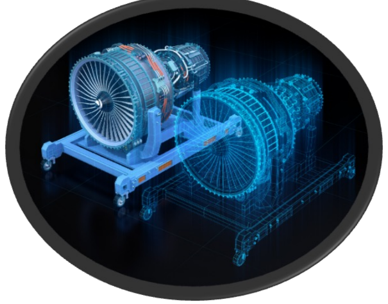
- Cage-based deformation



+



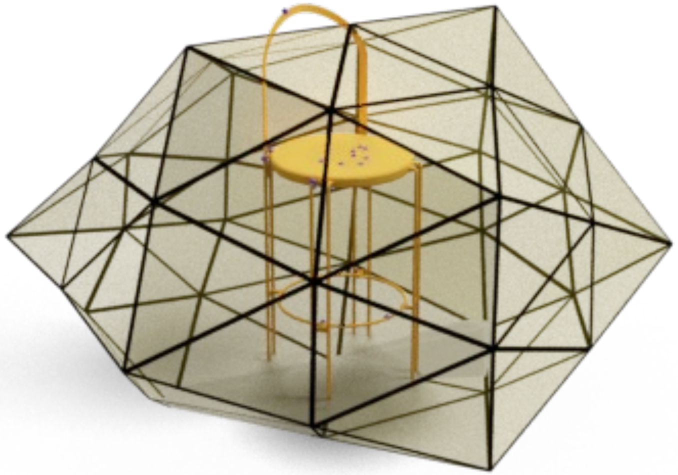
+



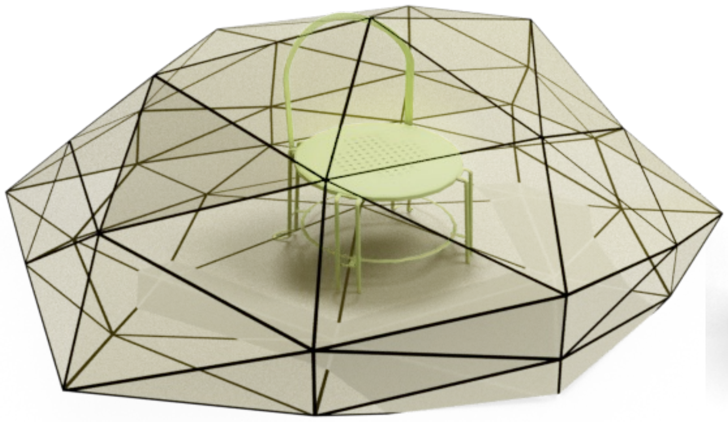
$$f_\theta : z_{\text{source}} \times z_{\text{target}} \rightarrow \boxed{\mathcal{C}_{\text{init}} \times \mathcal{C}_{\text{deformed}}} \rightarrow \boxed{\text{MVC}} \rightarrow \mathbb{R}^3 \rightarrow \mathbb{R}^3$$

Predict cage parameters with a neural network

Use Cage-Based Deformation to define the map



Init Cage



Deformed Cage

Cage-free Gradient Domain Deformation

- Hard to learn cages for complex shapes
- Cage-based deformation is not mesh-specific – it just maps the volume
- Reminder: learning a map directly is prone to noise – hard to preserve details

$$f_\theta : z_{\text{source}} \times z_{\text{target}} \times \mathbb{R}^3 \rightarrow \cancel{\mathbb{R}^3}\ \mathbb{R}^{3\times 3}$$

Predict a matrix

Cage-free Gradient Domain Deformation

- Hard to learn cages for complex shapes
- Cage-based deformation is not mesh-specific – it just maps the volume
- Reminder: learning a map directly is prone to noise – hard to preserve details

$$f_\theta : z_{\text{source}} \times z_{\text{target}} \times \mathbb{R}^3 \to \cancel{\mathbb{R}^3}\ \mathbb{R}^{3\times 3}$$

Predict a matrix

$$\downarrow \pi$$

$$\mathbb{R}^{3\times 2}$$

Project to Jacobian

Cage-free Gradient Domain Deformation

- Hard to learn cages for complex shapes
- Cage-based deformation is not mesh-specific – it just maps the volume
- Reminder: learning a map directly is prone to noise – hard to preserve details

$$f_\theta : z_{\text{source}} \times z_{\text{target}} \times \underset{\mathcal{S}}{\cancel{\mathbb{R}^3}} \rightarrow \cancel{\mathbb{R}^3}\, \mathbb{R}^{3\times 3}$$

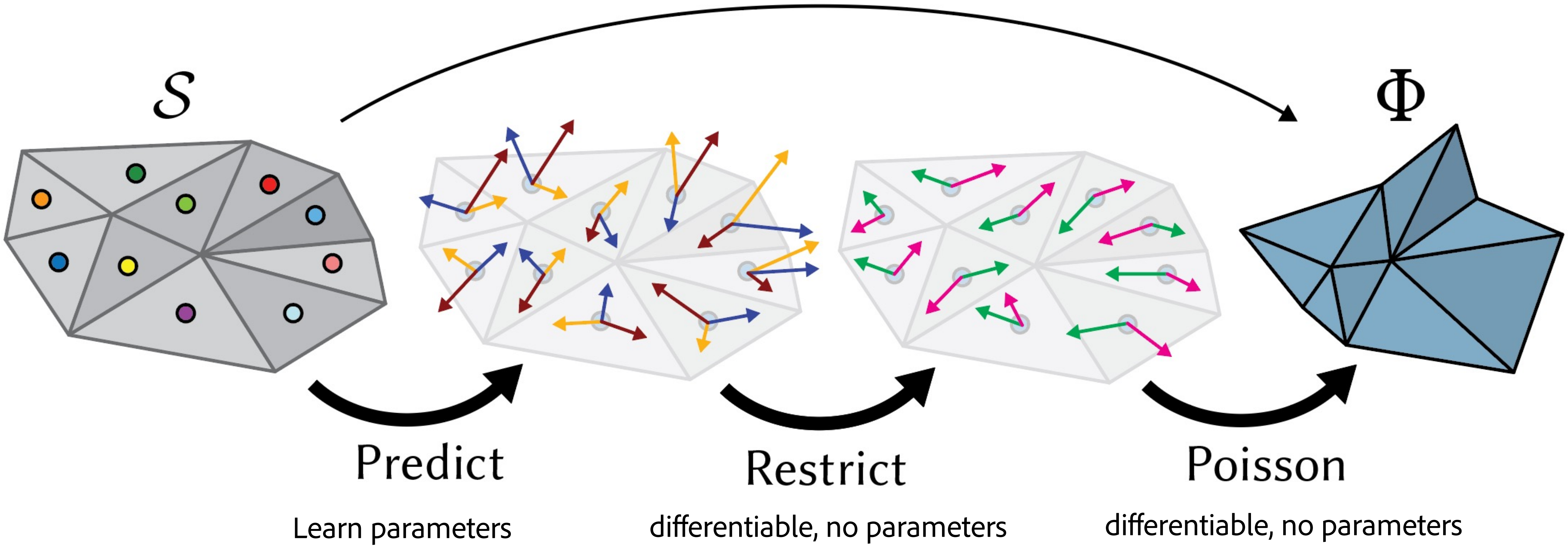
Use points on the surface (triangle centroids, Intrinsic features)

Predict a matrix

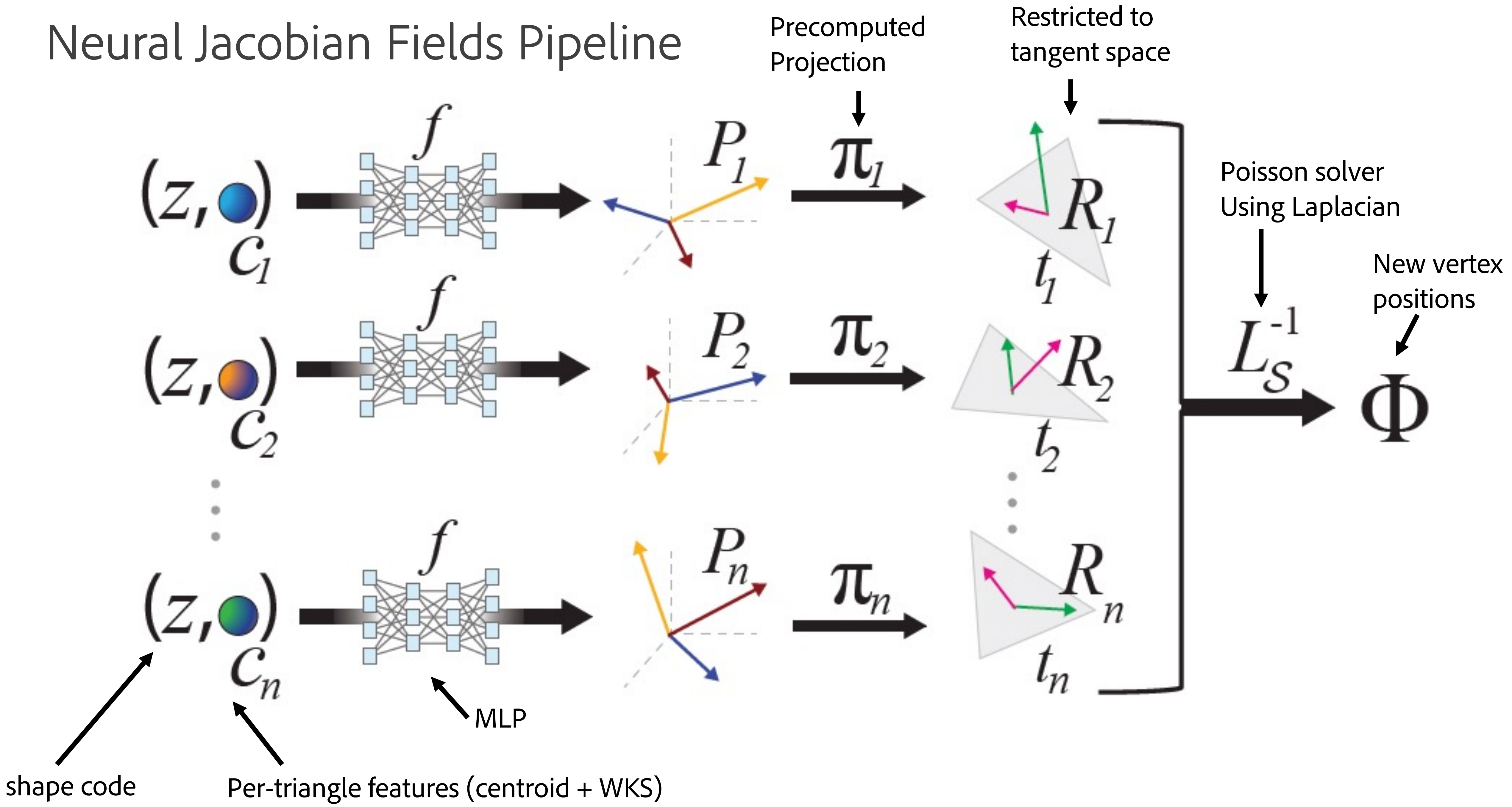
$$\mathbb{R}^{3\times 3} \Big\downarrow \pi \quad \mathbb{R}^{3\times 2}$$

Project to Jacobian

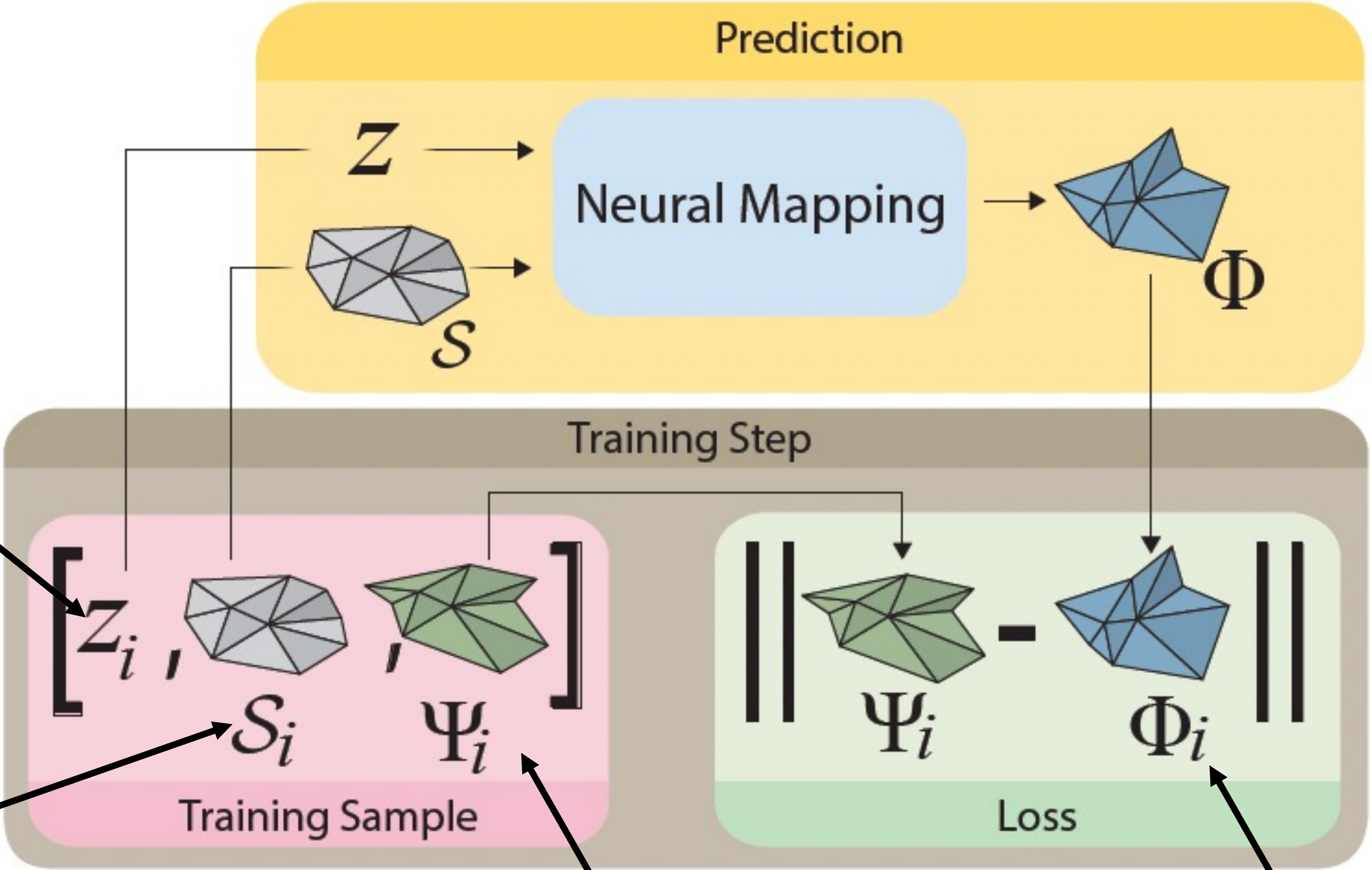
Neural Jacobian Fields Pipeline



Neural Jacobian Fields Pipeline



Training Neural Jacobian Fields



Shape & triangle codes

Source shape

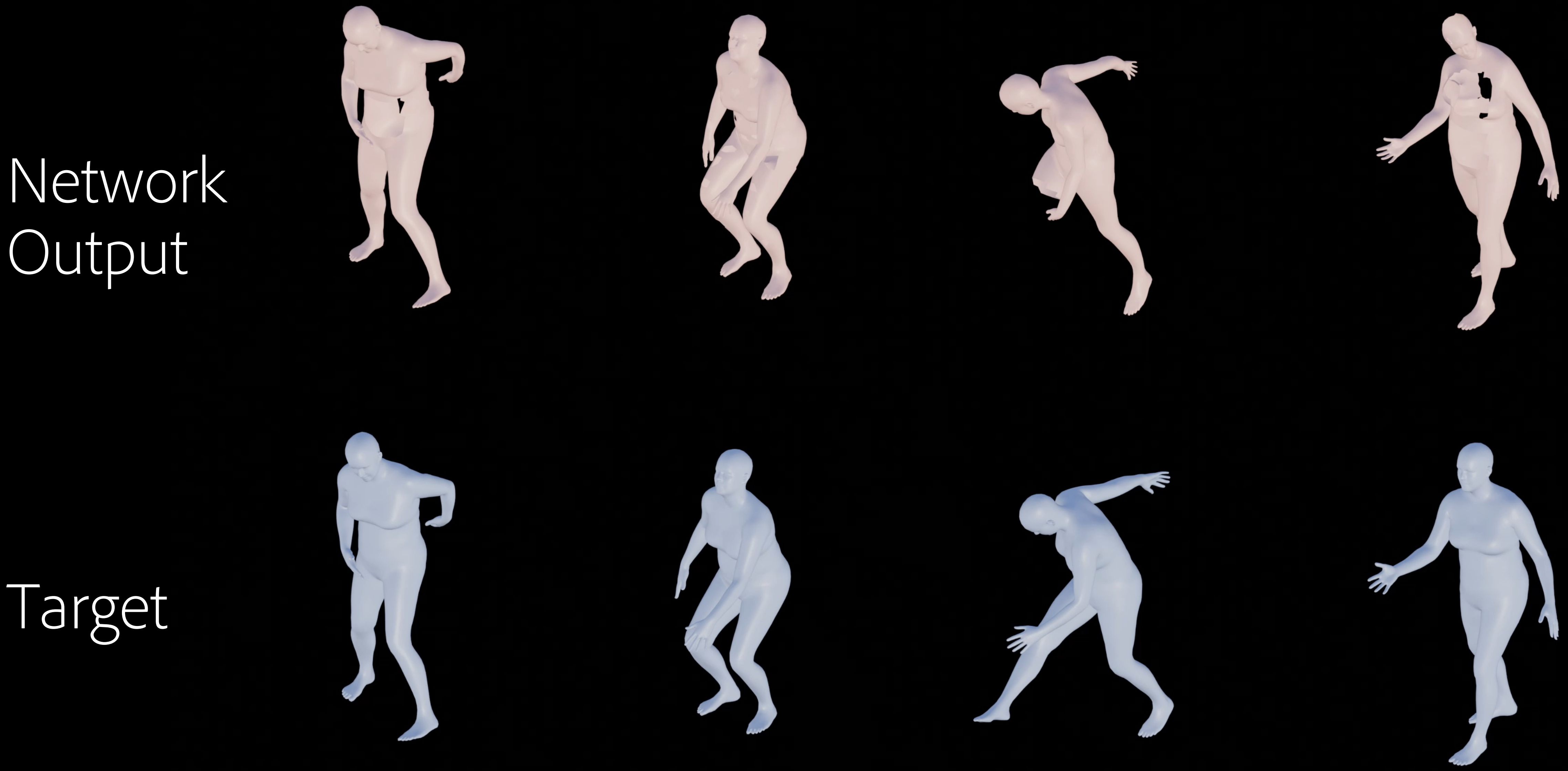
Target shape (GT)

Prediction (after Poisson)

Application: Deformation Transfer

Only trained on humans, no extra input was needed for Big Buck Bunny

Partial Registration

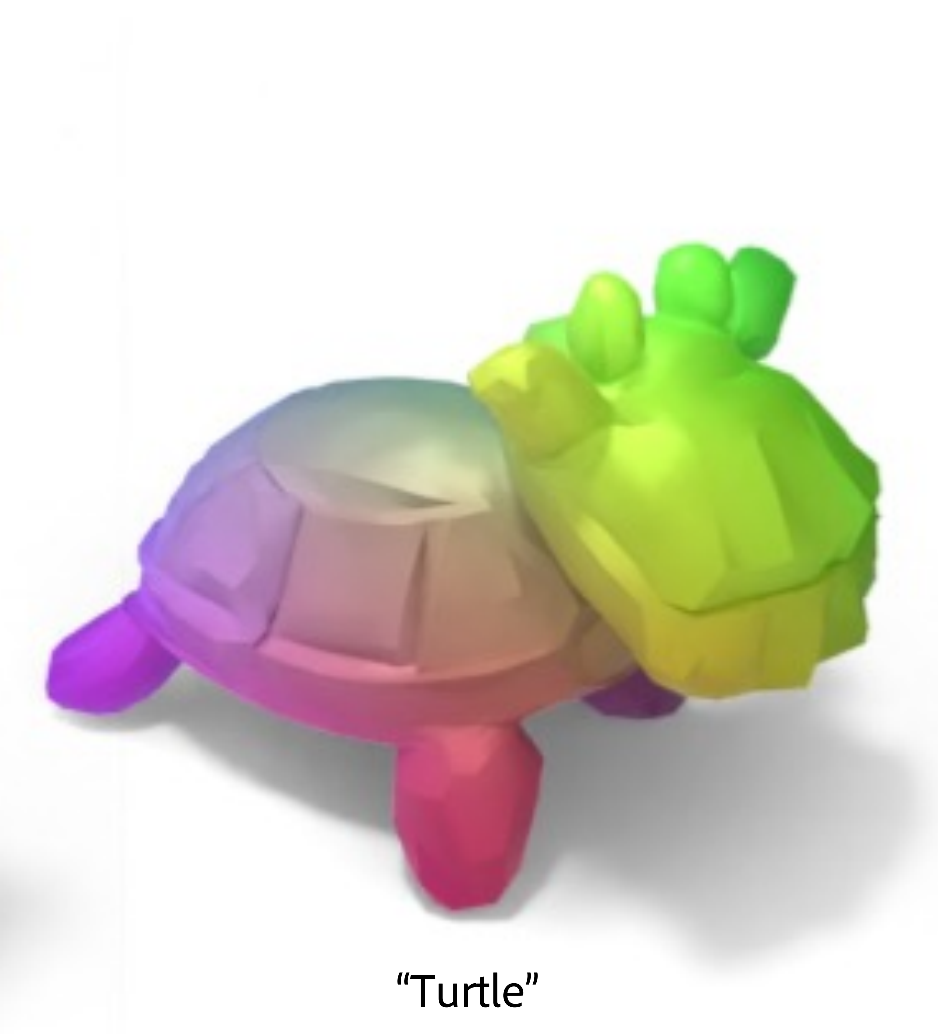


Morphing

Deformation with Text Guidance

Higher-level guidance for mesh deformation

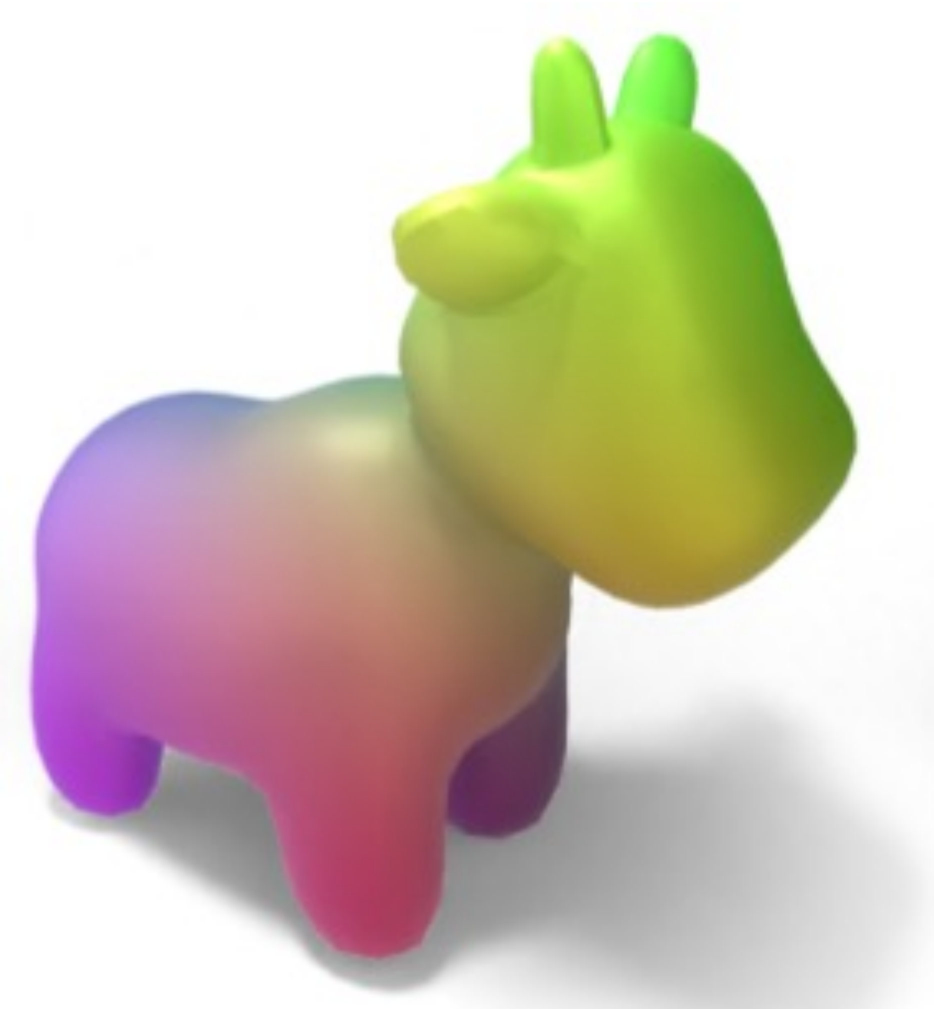
Source



"Turtle"

Deformation with Text Guidance

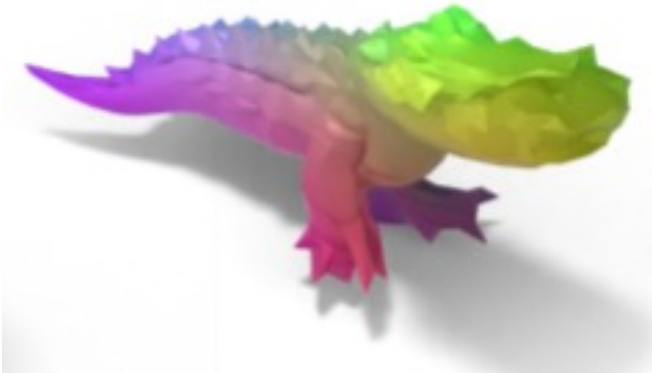
Higher-level guidance for mesh deformation



Source

“Turtle”

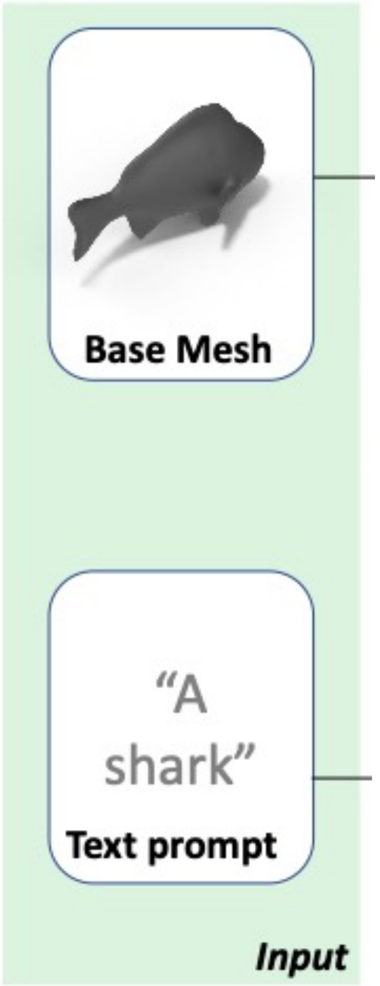
“Giraffe”



“Alligator”

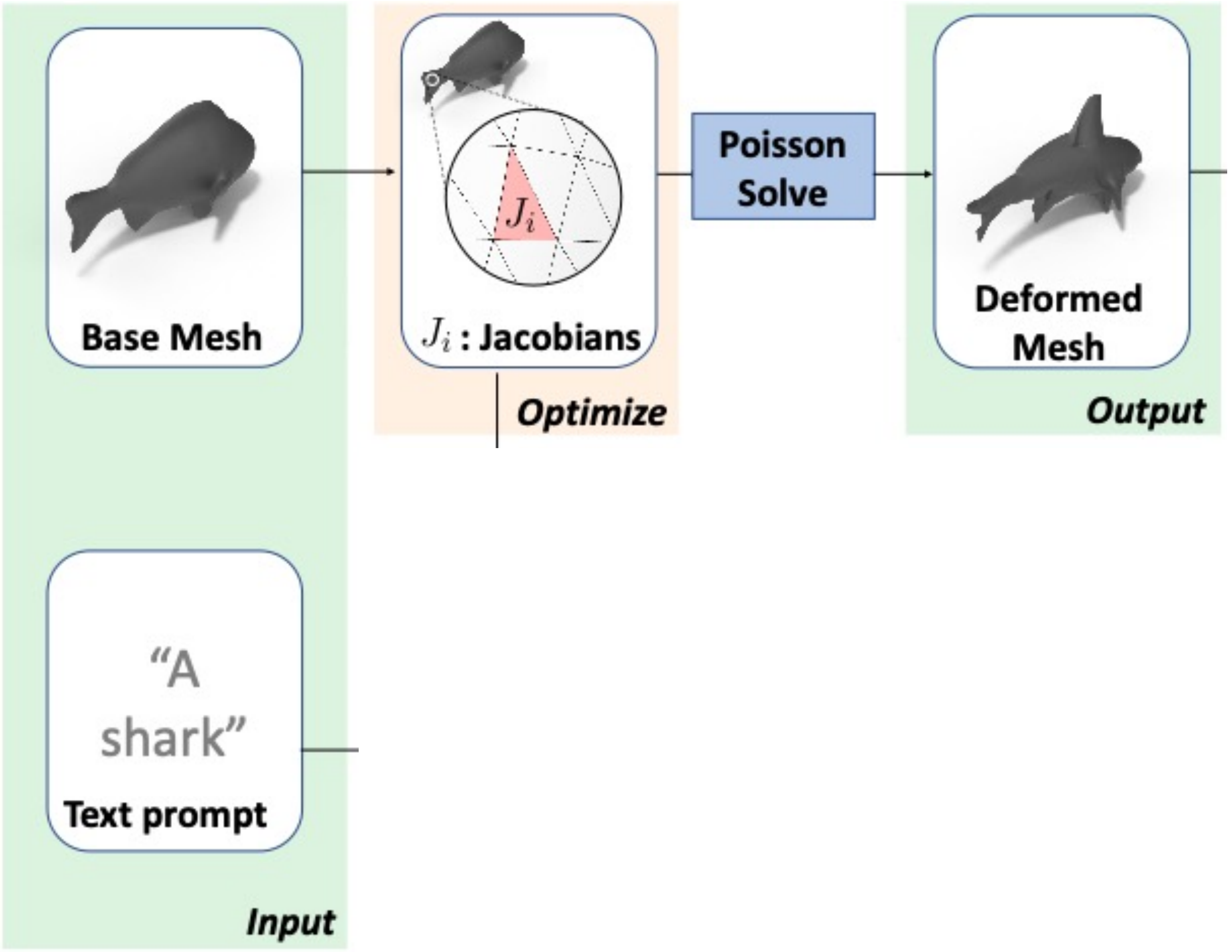
Deformation with Text Guidance

Optimize Jacobians to minimize CLIP-similarity loss



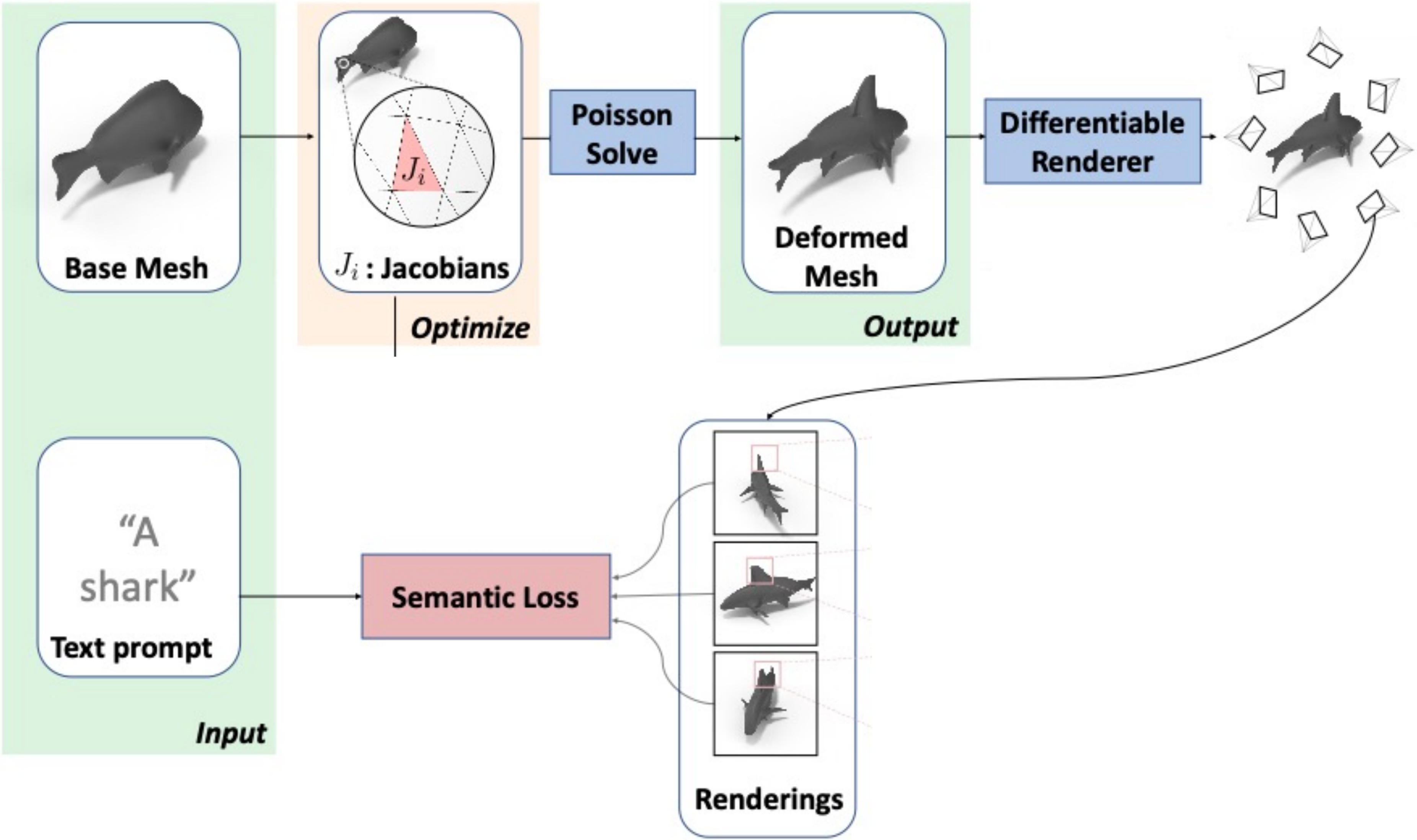
Deformation with Text Guidance

Optimize Jacobians to minimize CLIP-similarity loss



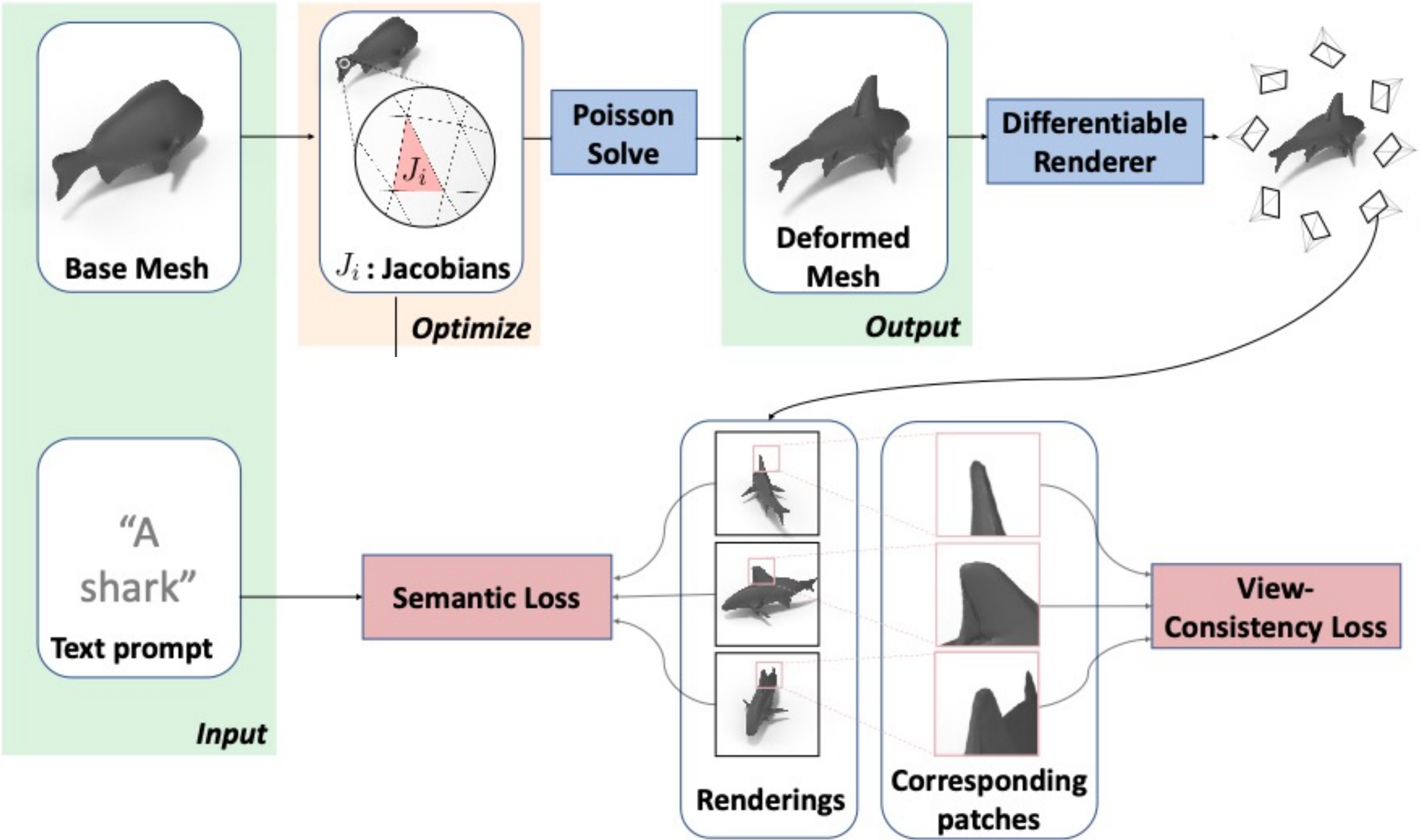
Deformation with Text Guidance

Optimize Jacobians to minimize CLIP-similarity loss



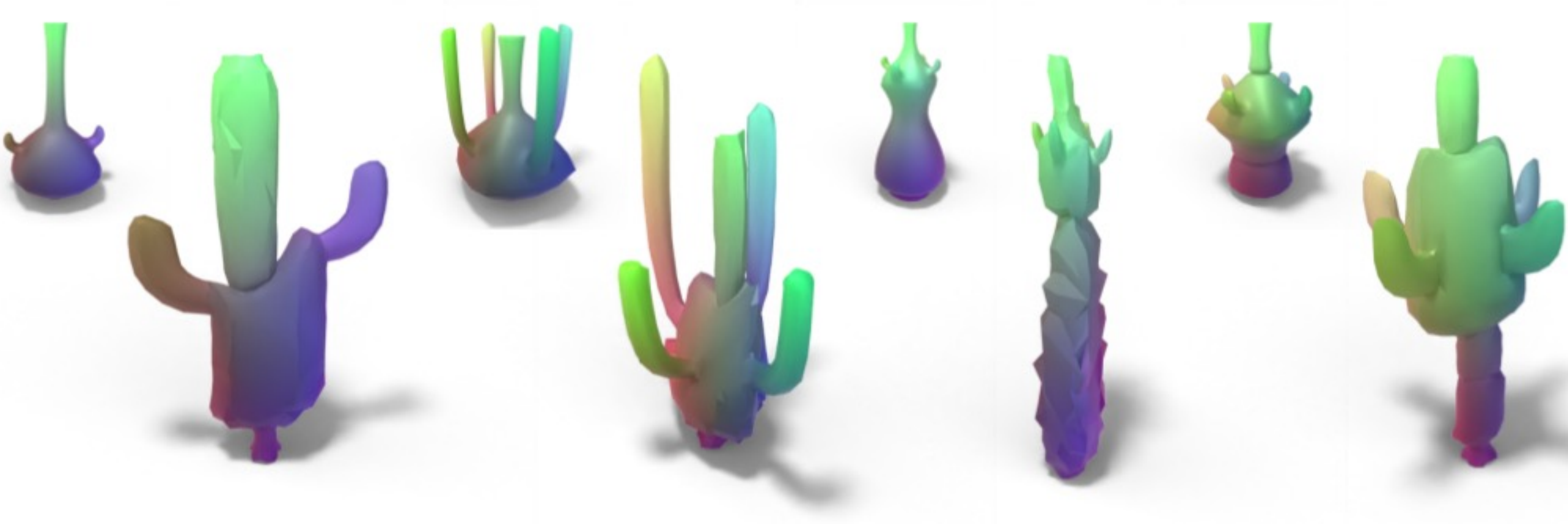
Deformation with Text Guidance

Optimize Jacobians to minimize CLIP-similarity loss



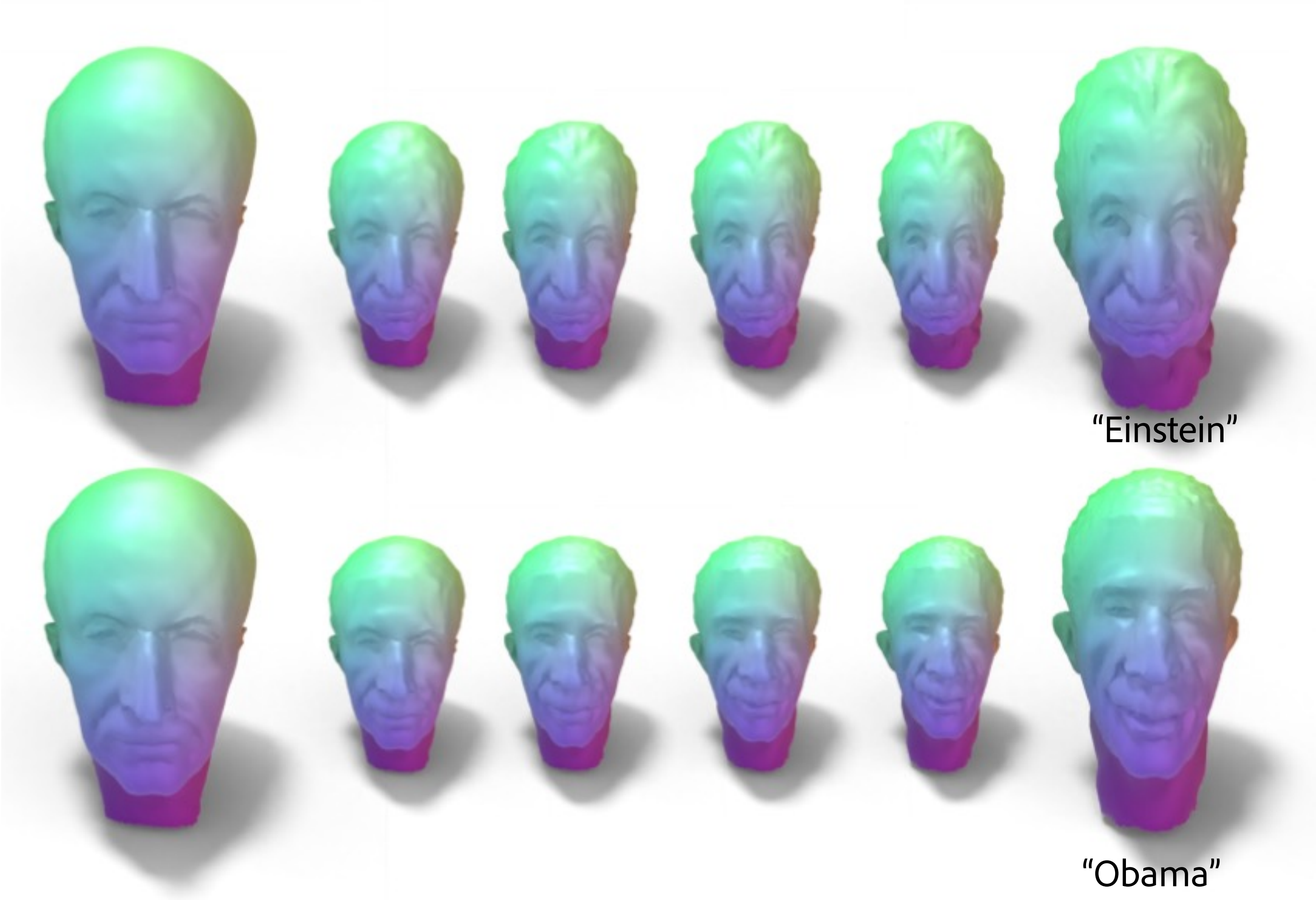
Deformation with Text Guidance

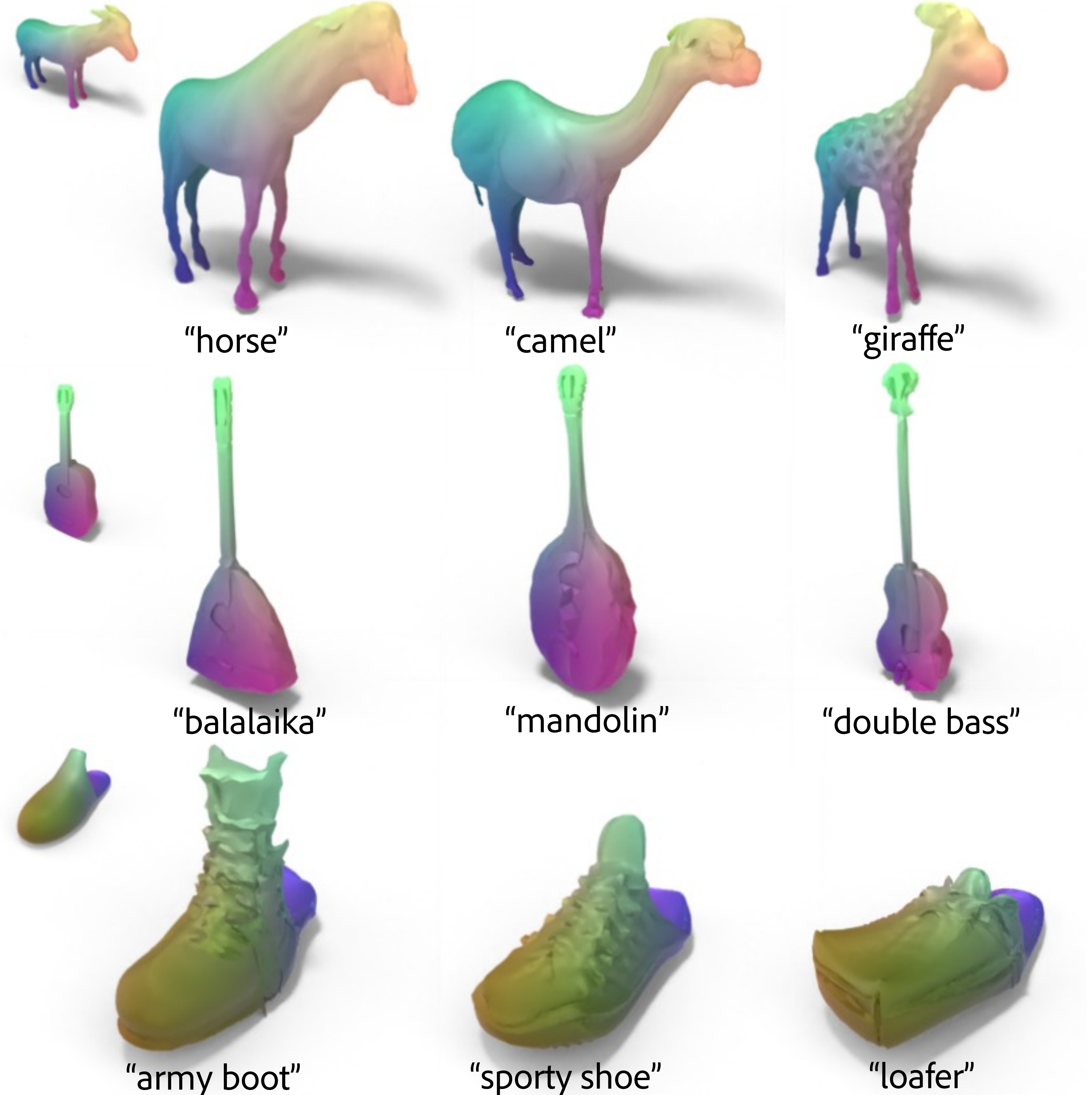
Convert vases to cactuses



Deformation with Text Guidance

Deforming faces





“horse”
“camel”
“giraffe”
“balalaika”
“mandolin”
“double bass”
“army boot”
“sporty shoe”
“loafer”

Neural Deformation Takeaways

Use classical Geometry Processing modules as layers in NN, e.g.:

- Cage-based deformation
- Deformation Jacobians
- Poisson solve

Neural Deformation Takeaways

Use classical Geometry Processing modules as layers in NN, e.g.:

- Cage-based deformation
- Deformation Jacobians
- Poisson solve

Neural Networks can:

- Optimize quickly by solving similar problems on training data
- Implicitly learn relations between related shapes during training
- Pre-trained visual networks (CLIP, Diffusion Models) offer a strong prior on the natural objects

Neural Deformation Takeaways

Use classical Geometry Processing modules as layers in NN, e.g.:

- Cage-based deformation
- Deformation Jacobians
- Poisson solve

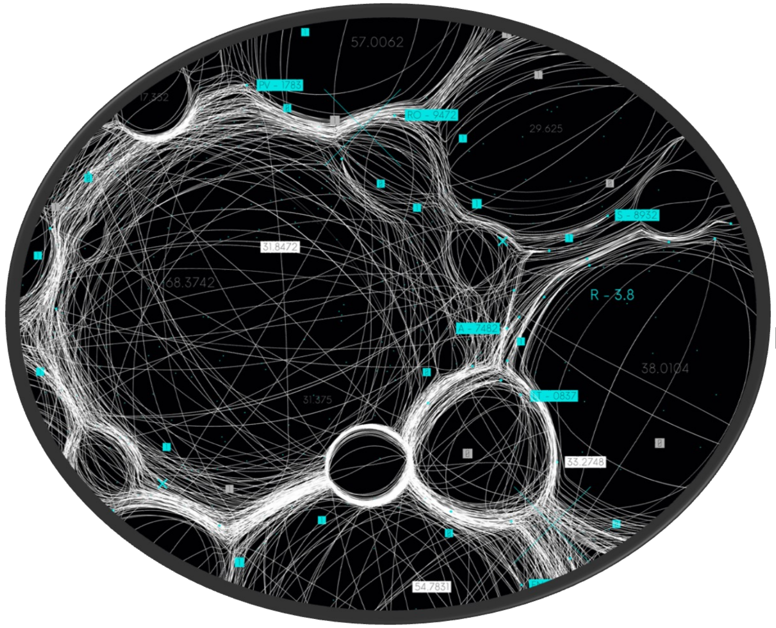
Neural Networks can:

- Optimize quickly by solving similar problems on training data
- Implicitly learn relations between related shapes during training
- Pre-trained visual networks (CLIP, Diffusion Models) offer a strong prior on the natural objects

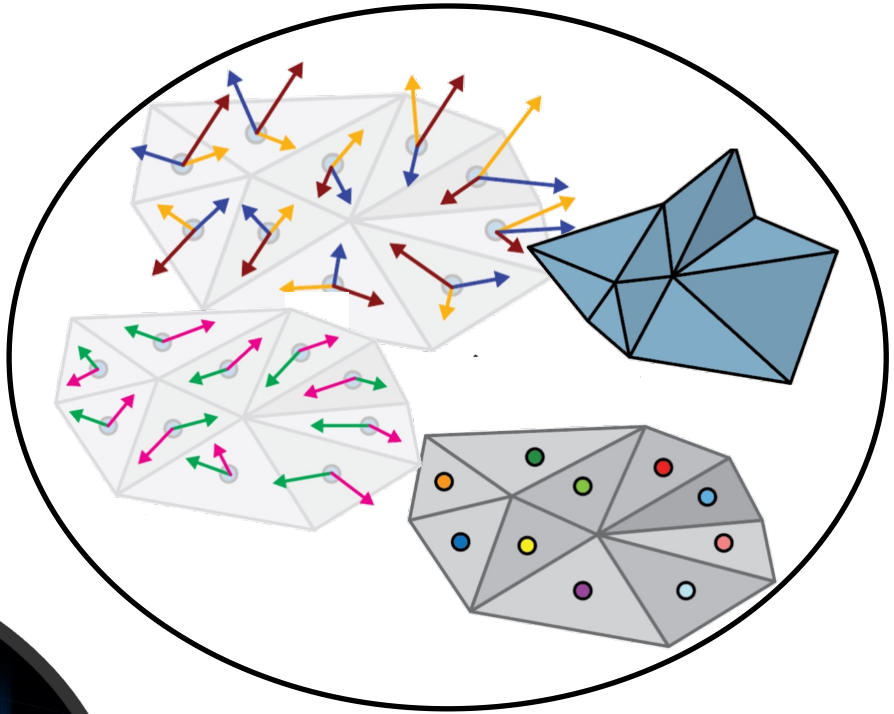
Questions for the Future Work

- What are the other interesting ways to parameterize shape variations?
- What new techniques and loss functions can we develop to interject pre-trained visual priors?
- Can we train 3D neural networks using a mix of 2D and 3D data? A mix of strong and weak supervision?

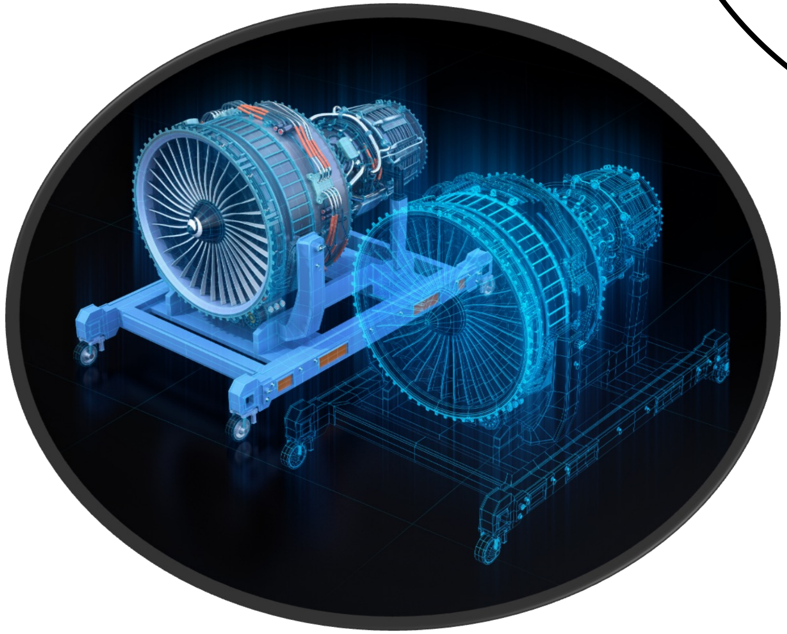
Machine Learning Helping Geometry Processing



Neural Networks



Geometry Processing



Meshes

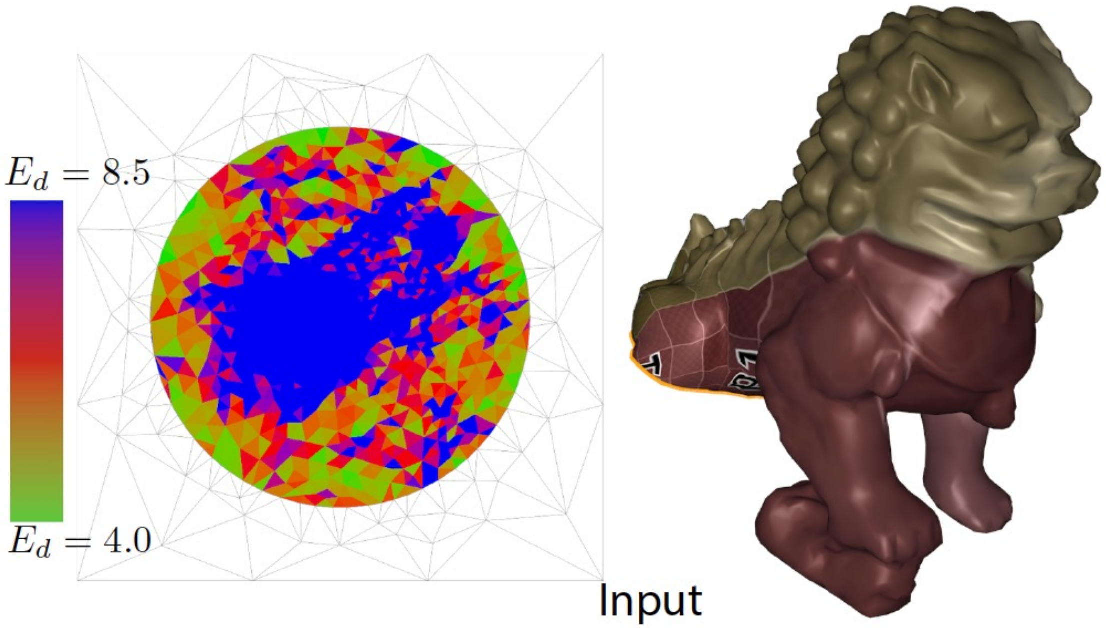
Deformation for Mesh Parameterization

Why parameterize?

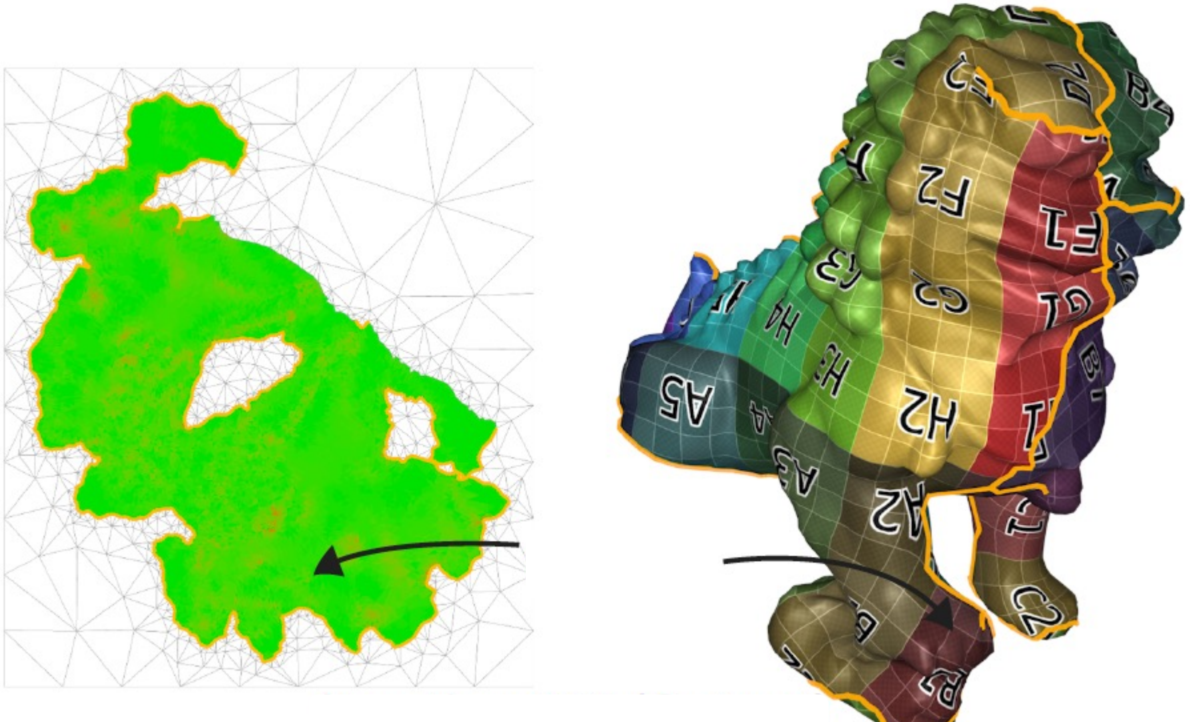
- Concisely store signals (e.g., materials) on surfaces
- Essential for most existing pipelines

What is a good parameterization?

- Small distortion (squares in 2D look like squares in 3D)
- Few discontinuities



Bad Parameterization



Better Parameterization

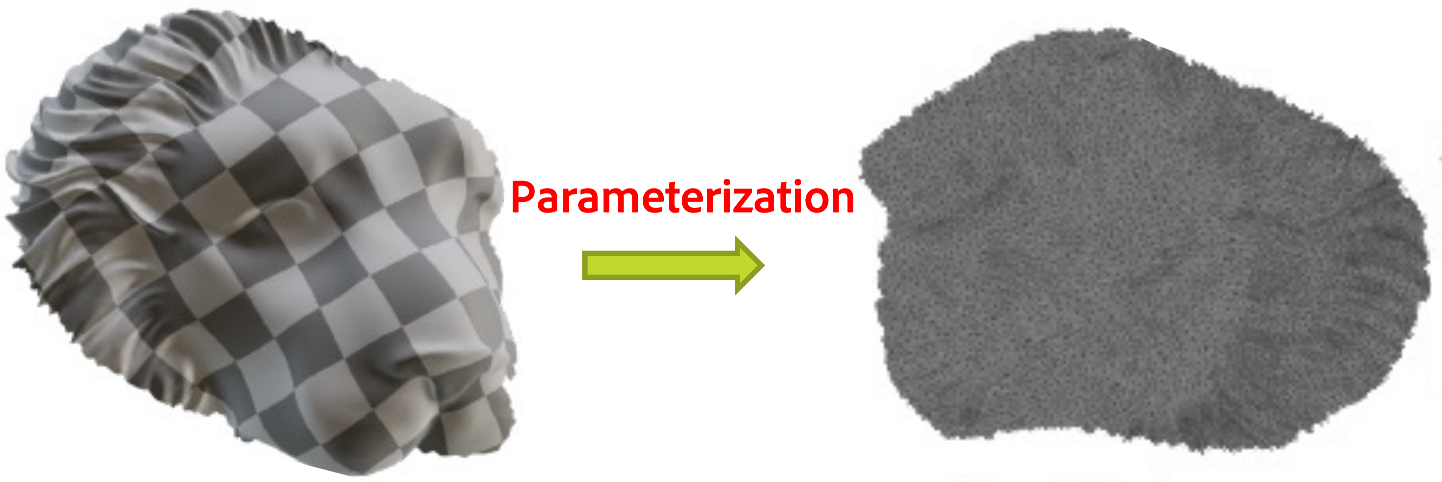
Deformation for Mesh Parameterization

Why parameterize with ML?

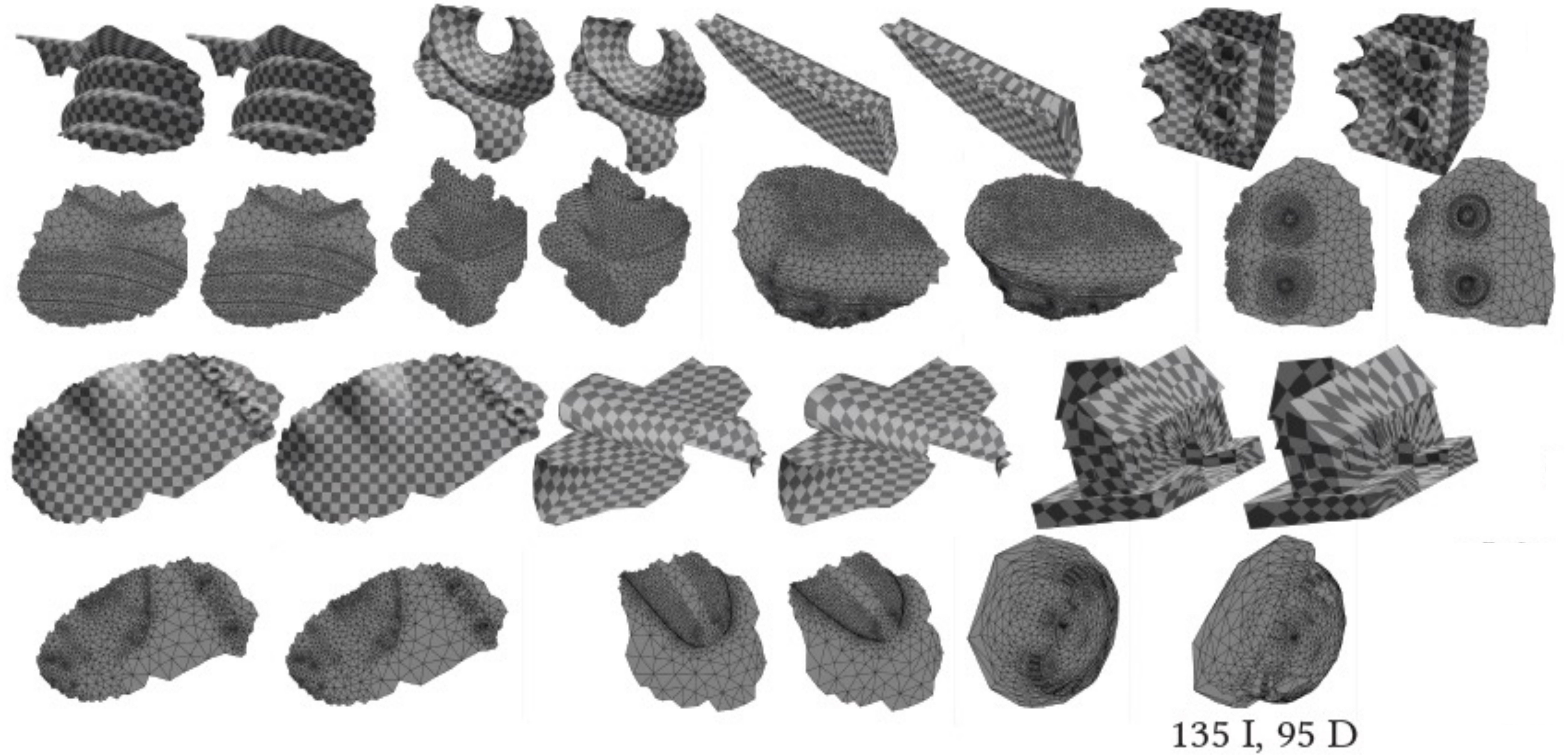
- Lots of local optima – classical optimization methods are slow and prone to being stuck
- Learn to mimic artists – hide distortion and seams in non-salient regions

How to parameterize with ML?

- Neural Jacobian Fields to map 3D to 2D
- Train with strong supervision using classical geometry optimization (SLIM)

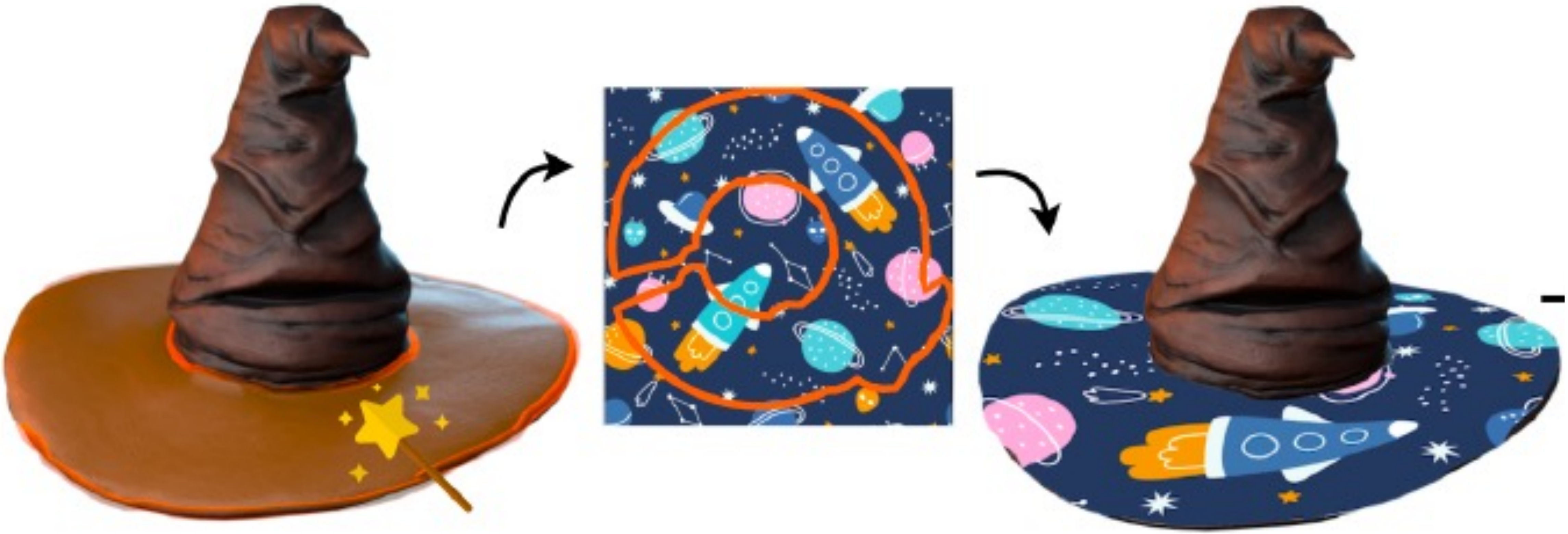


Deformation for Mesh Parameterization



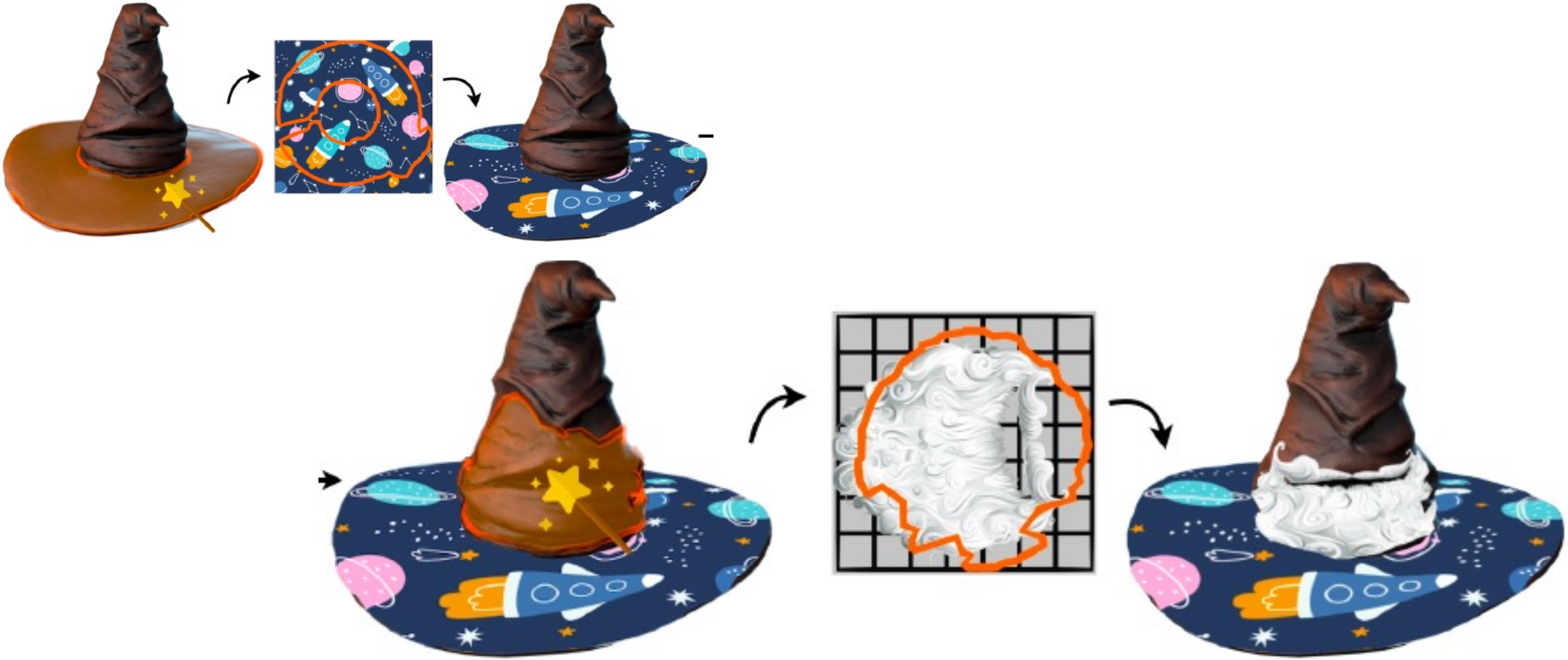
Segmentation for Parameterization

Given a selected point, find maximal segment that can be parameterized with little distortion

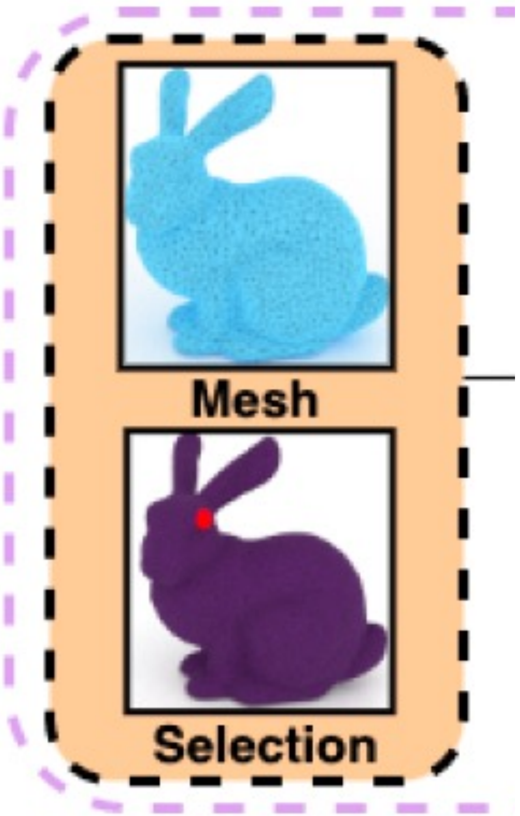


Segmentation for Parameterization

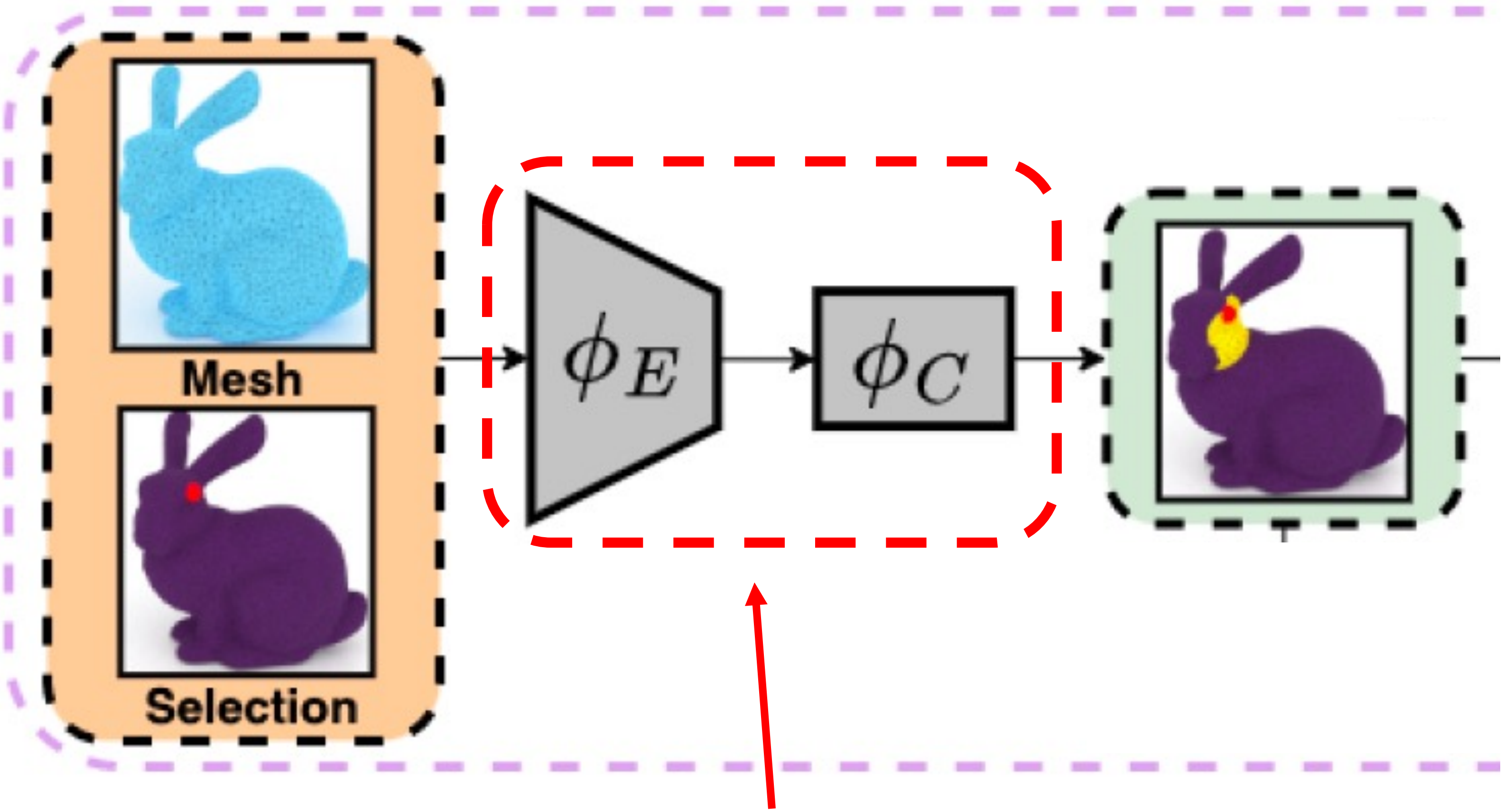
Given a selected point, find maximal segment that can be parameterized with little distortion



Segmentation for Parameterization

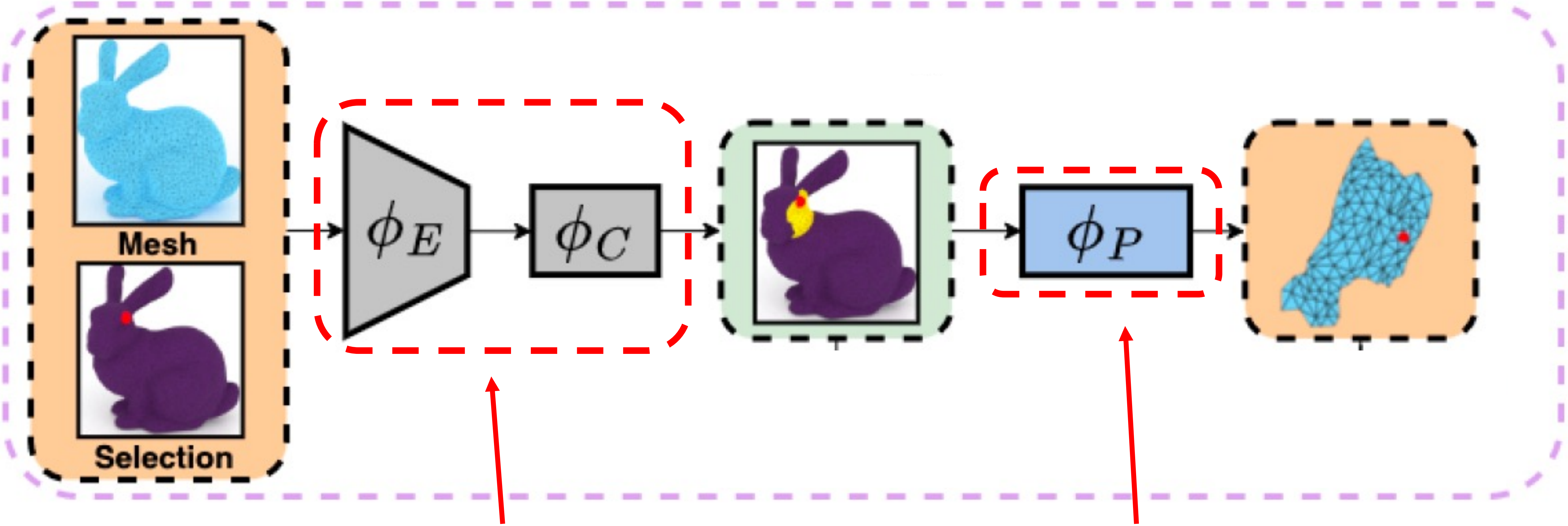


Segmentation for Parameterization



Train-able Segmentation Network
(MeshCNN)

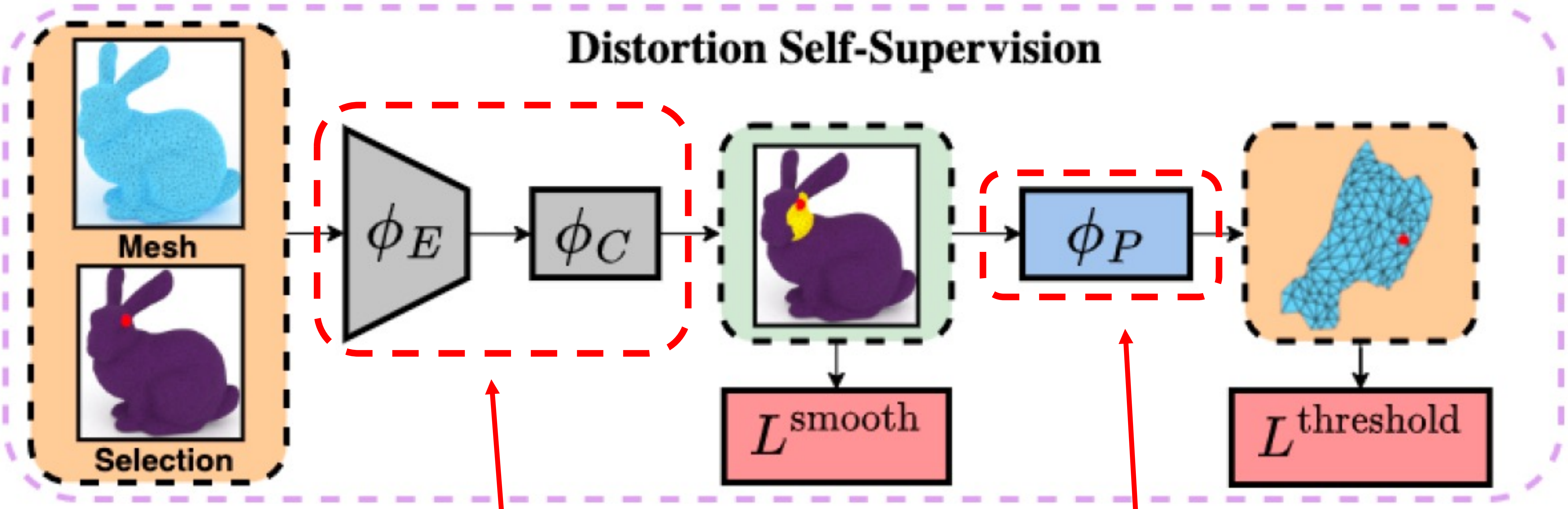
Segmentation for Parameterization



Train-able Segmentation Network
(MeshCNN)

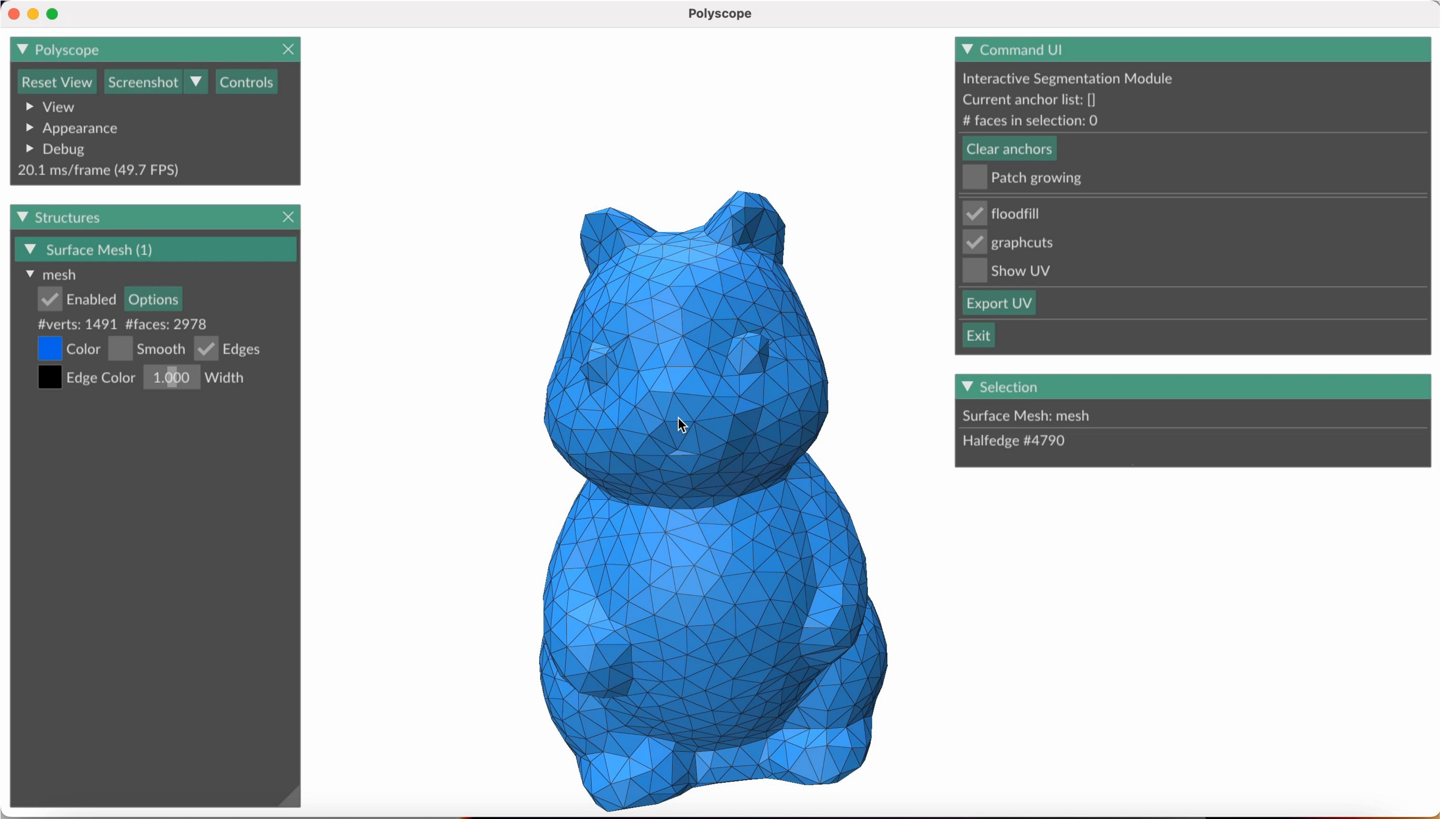
Differentiable Parameterization Layer
(wLSCM)

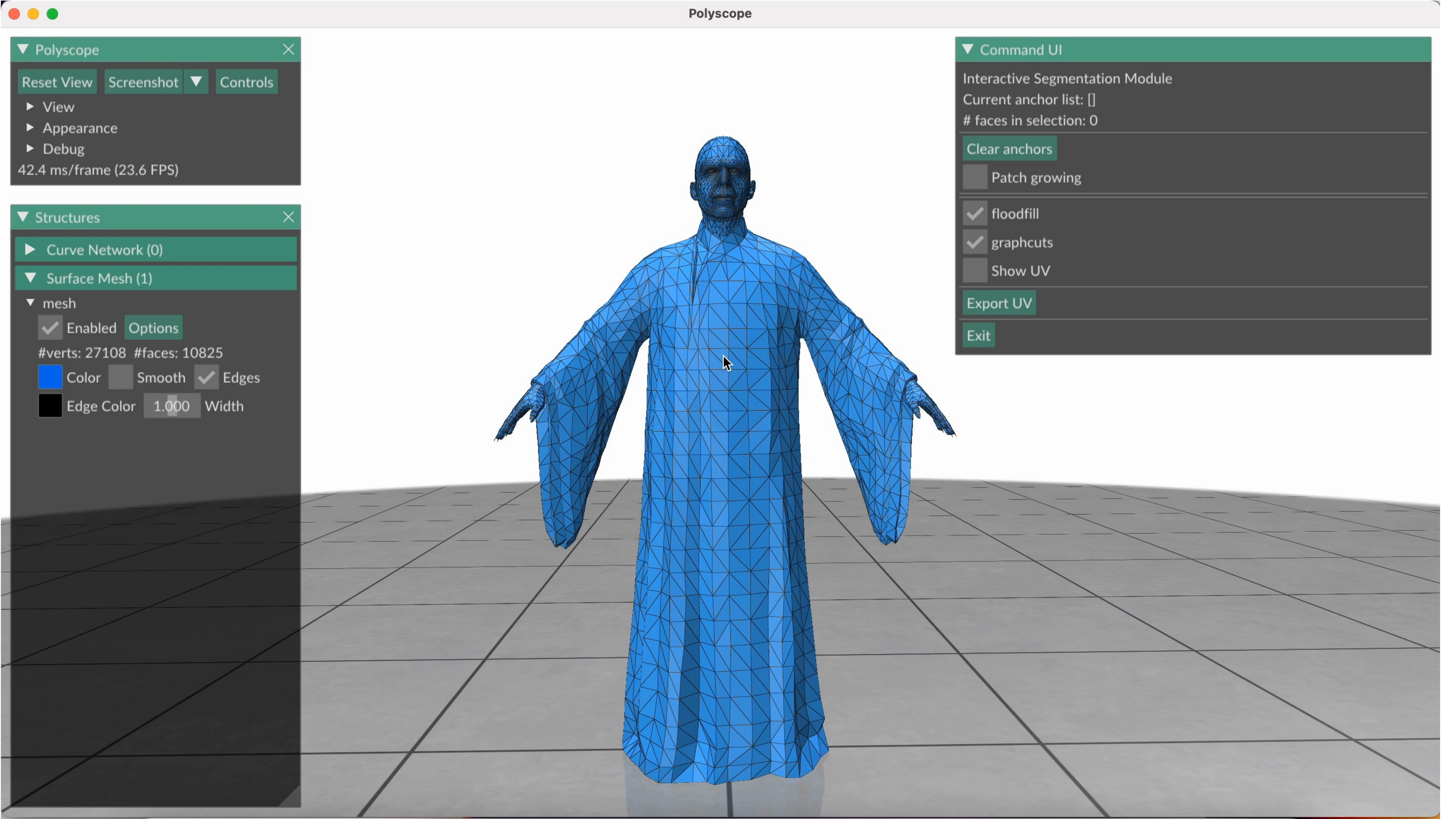
Segmentation for Parameterization

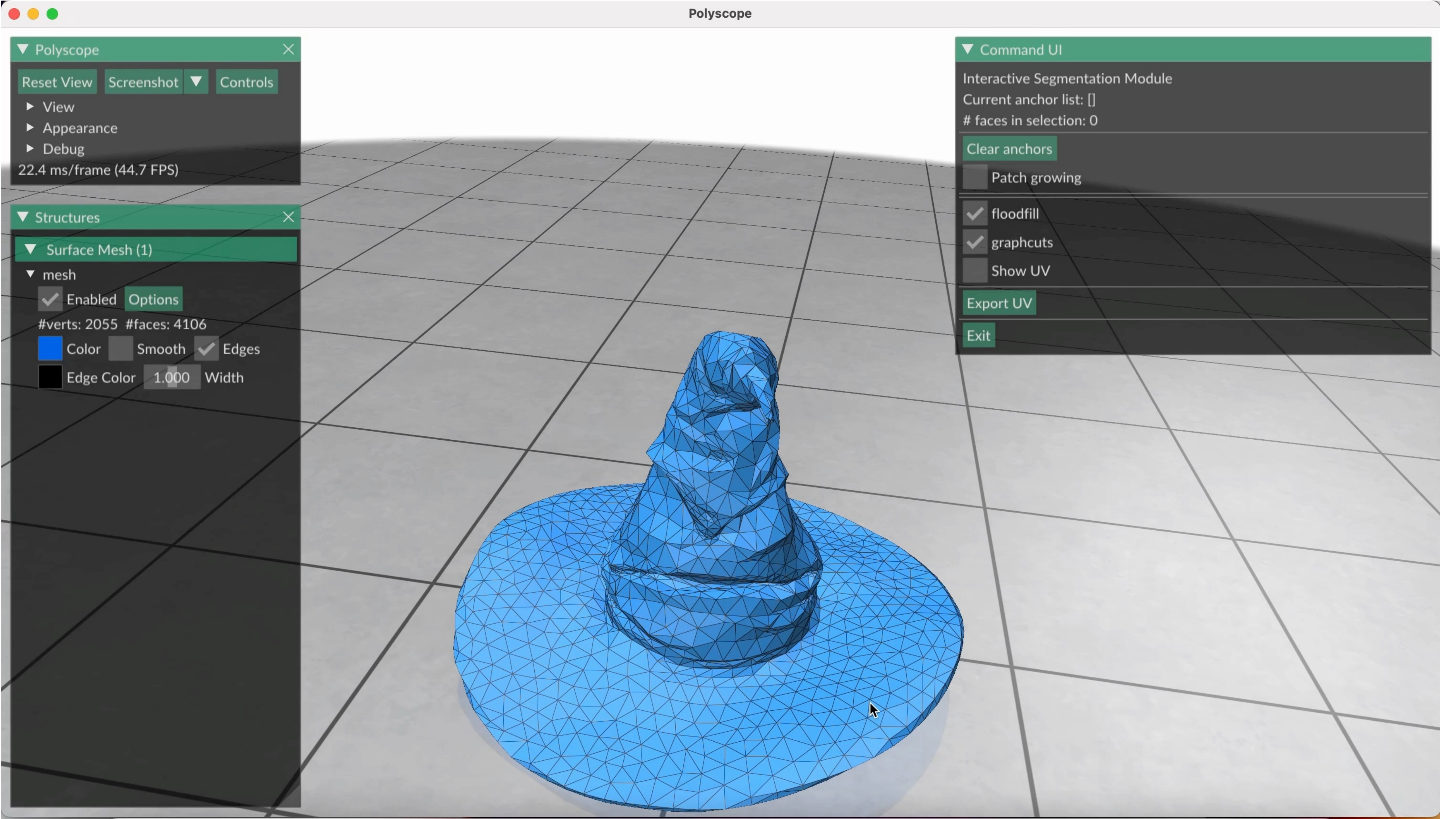


Train-able Segmentation Network
(MeshCNN)

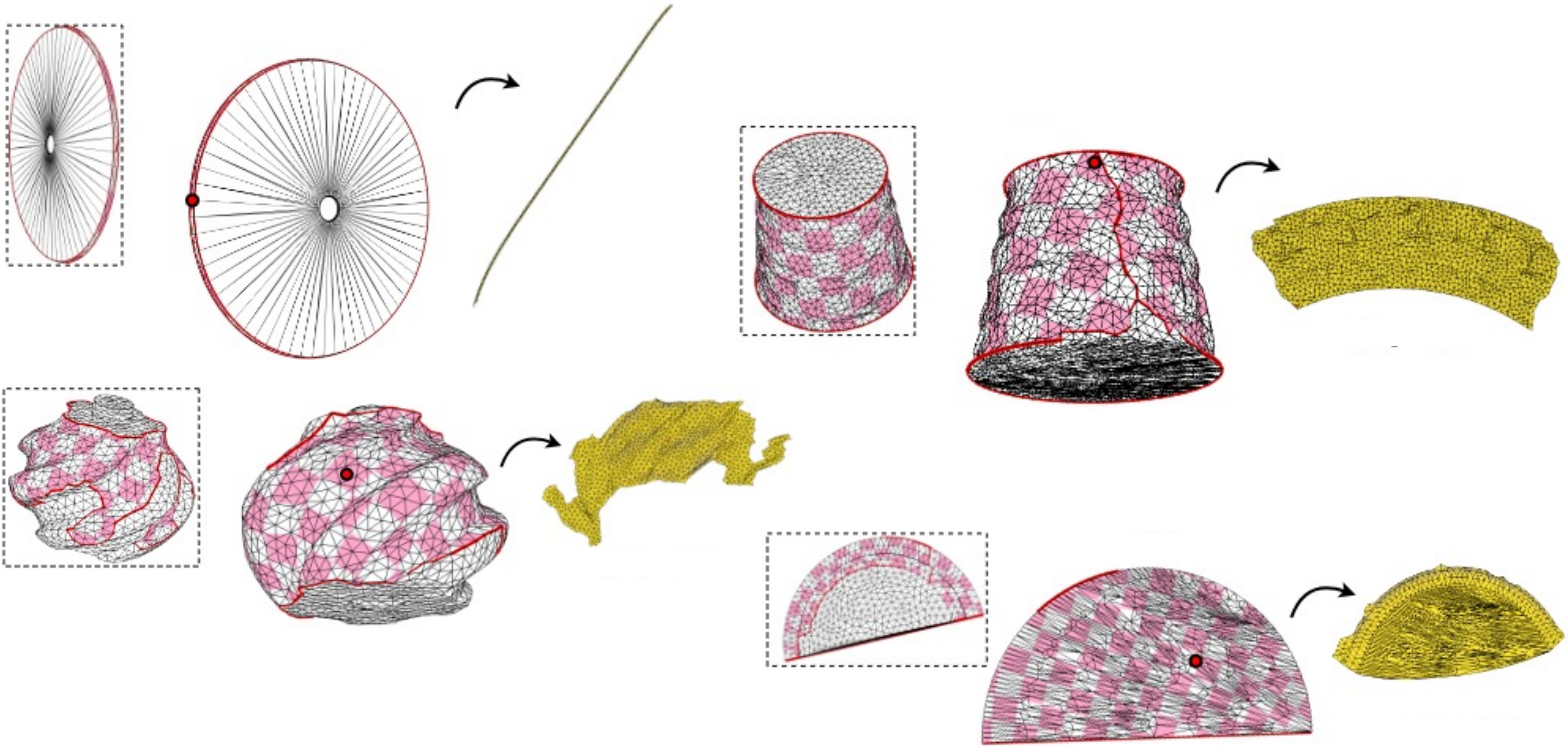
Differentiable Parameterization Layer
(wLSCM)


Polyscope
Polyscope
Reset View
Screenshot
Controls
View
Appearance
Debug
20.1 ms/frame (49.7 FPS)
Structures
Surface Mesh (1)
mesh
Enabled
Options
#verts: 1491 #faces: 2978
Color
Smooth
Edges
Edge Color
1.000
Width
Command UI
Interactive Segmentation Module
Current anchor list: []
faces in selection: 0
Clear anchors
Patch growing
floodfill
graphcuts
Show UV
Export UV
Exit
Selection
Surface Mesh: mesh
Halfedge #4790


Polyscope
Polyscope
Reset View
Screenshot
Controls
View
Appearance
Debug
42.4 ms/frame (23.6 FPS)
Structures
Curve Network (0)
Surface Mesh (1)
mesh
Enabled
Options
#verts: 27108 #faces: 10825
Color
Smooth
Edges
Edge Color
1.000
Width
Command UI
Interactive Segmentation Module
Current anchor list: []
faces in selection: 0
Clear anchors
Patch growing
floodfill
graphcuts
Show UV
Export UV
Exit


Polyscope
Polyscope
Reset View
Screenshot
Controls
View
Appearance
Debug
22.4 ms/frame (44.7 FPS)
Structures
Surface Mesh (1)
mesh
Enabled
Options
#verts: 2055 #faces: 4106
Color
Smooth
Edges
Edge Color
1.000
Width
Command UI
Interactive Segmentation Module
Current anchor list: []
faces in selection: 0
Clear anchors
Patch growing
floodfill
graphcuts
Show UV
Export UV
Exit

Segmentation for Parameterization



Neural Parameterization Takeaways

Use classical Geometry Processing modules as layers in NN, e.g.:

- Deformation Jacobians + Poisson solve
- Least-Squares Conformal Maps

Neural Networks can:

- Optimize quickly by solving similar problems on training data
- Implicitly learn relations between related shapes during training

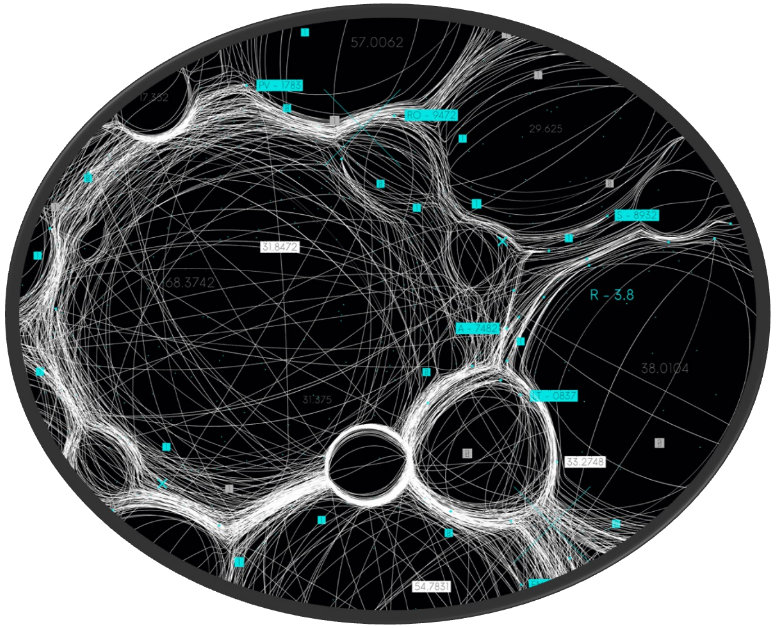
Questions for the Future Work

- Can pre-trained visual priors help improve parameterization?
- How do we represent discontinuities for global parameterization?

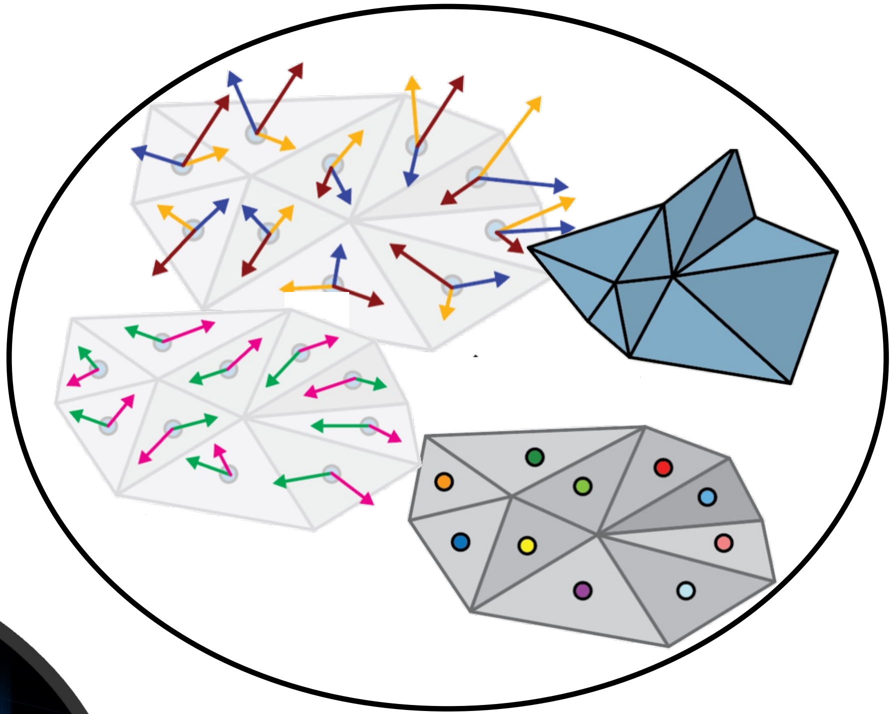
OptCuts: Joint Optimization of Surface Cuts and Parameterization

Submission ID: 243

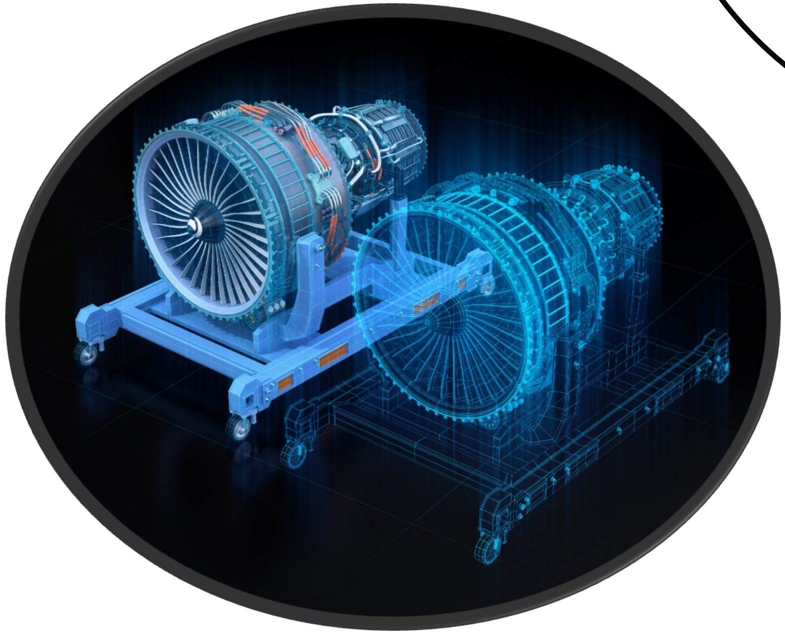
Machine Learning Helping Geometry Processing



Neural Networks



Geometry Processing



Meshes

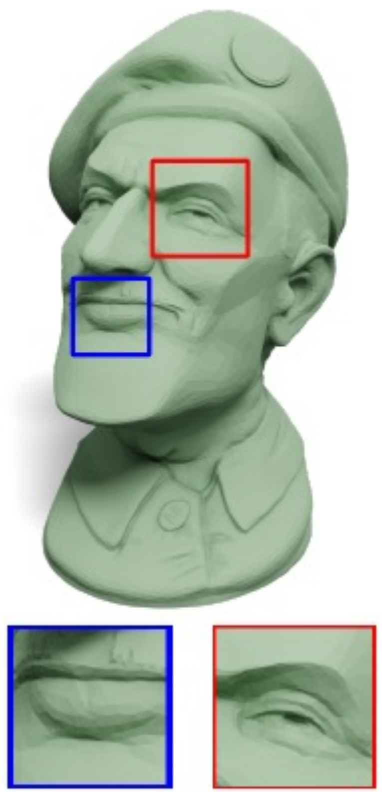
~~Neural~~ Progressive Meshes

Idea: Transmit 3D data progressively in a coarse-to-fine fashion

Hoppe 1996

Neural Progressive Meshes

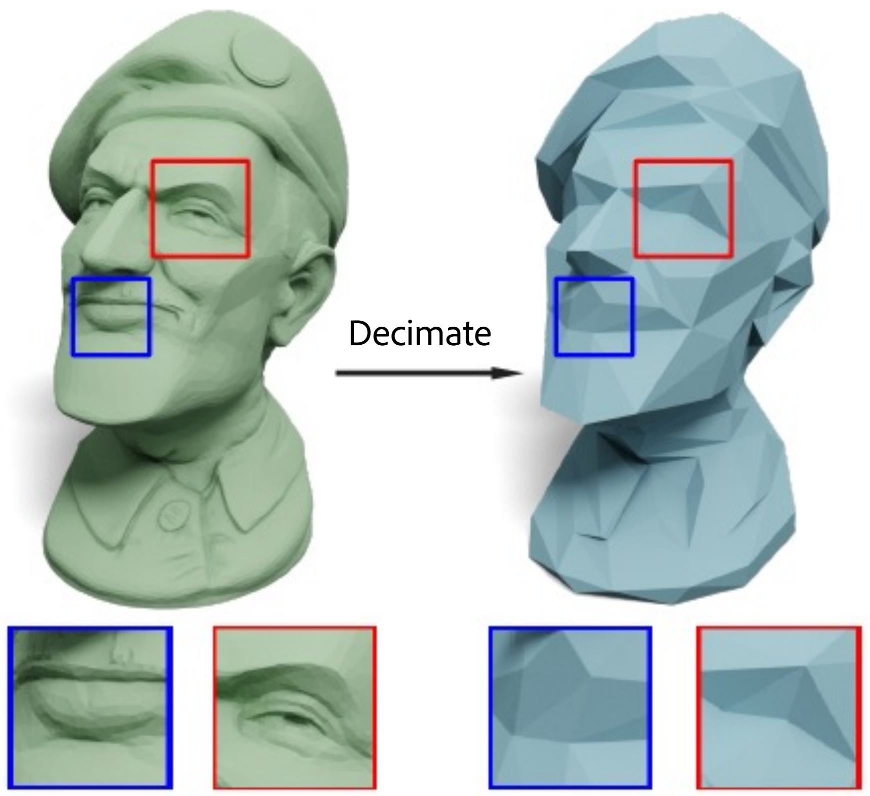
Learn a latent space of progressive features that encode geometric details



Ground truth

Neural Progressive Meshes

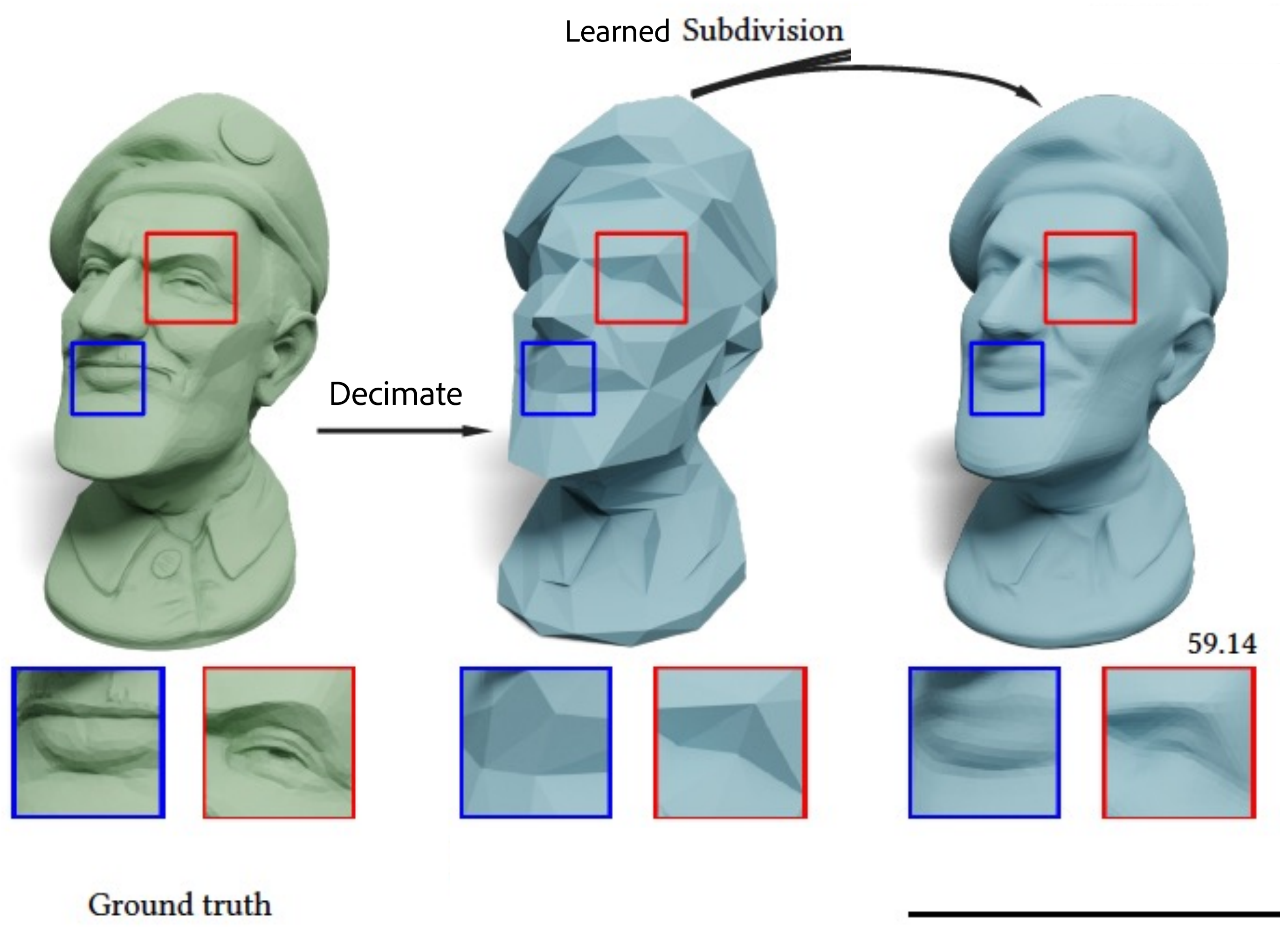
Learn a latent space of progressive features that encode geometric details



Ground truth

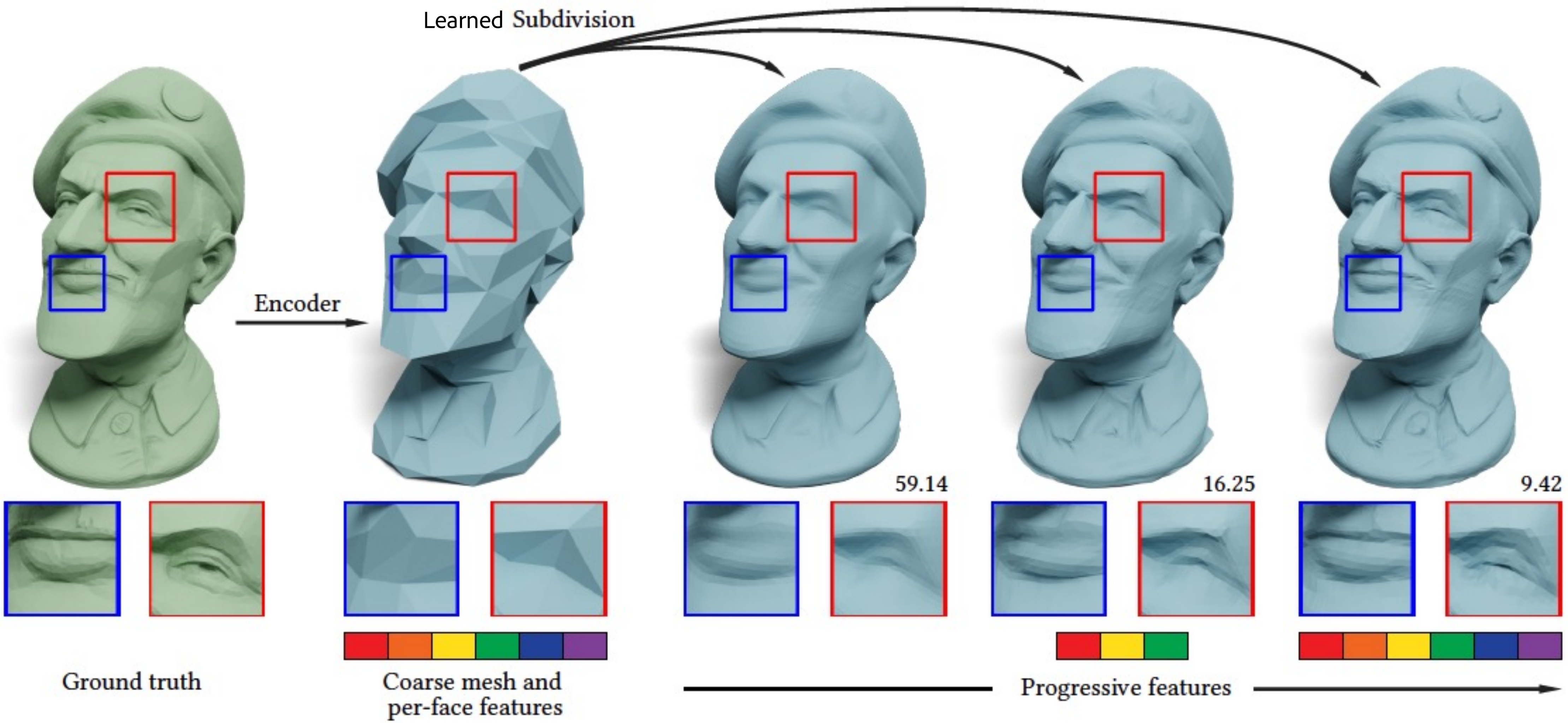
Neural Progressive Meshes

Learn a latent space of progressive features that encode geometric details

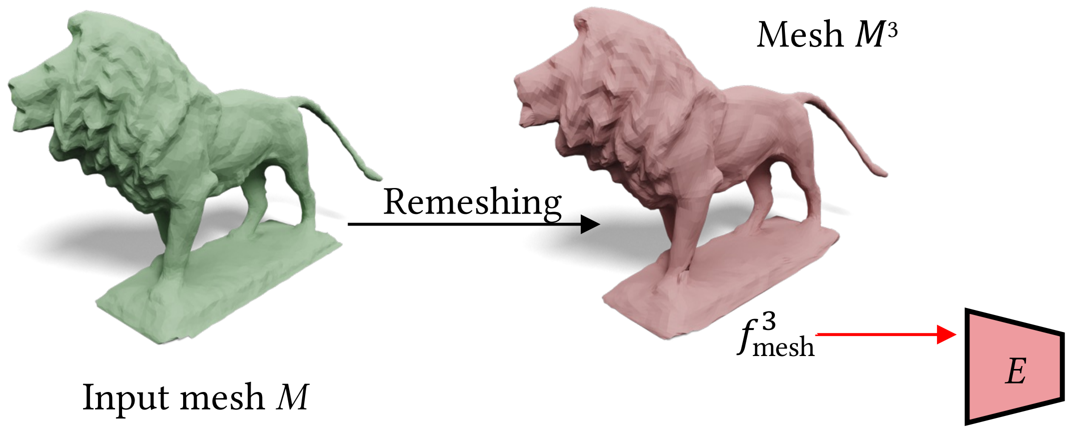


Neural Progressive Meshes

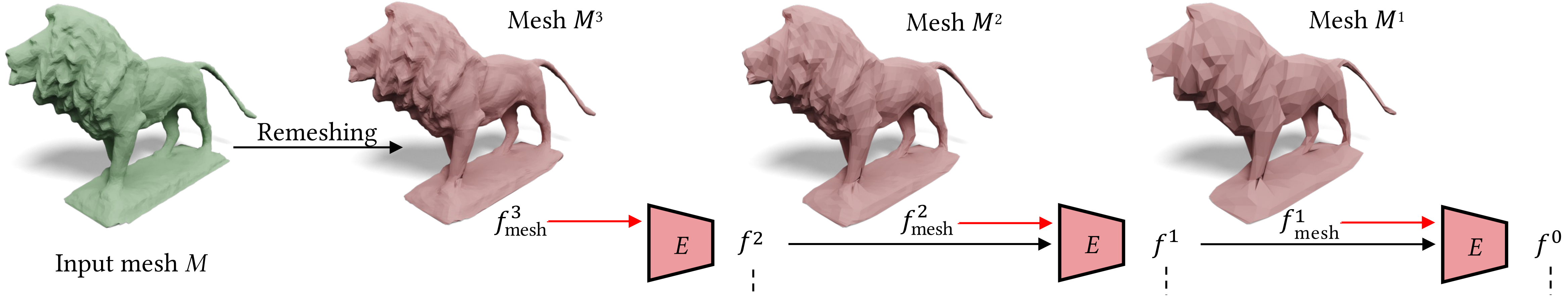
Learn a latent space of progressive features that encode geometric details



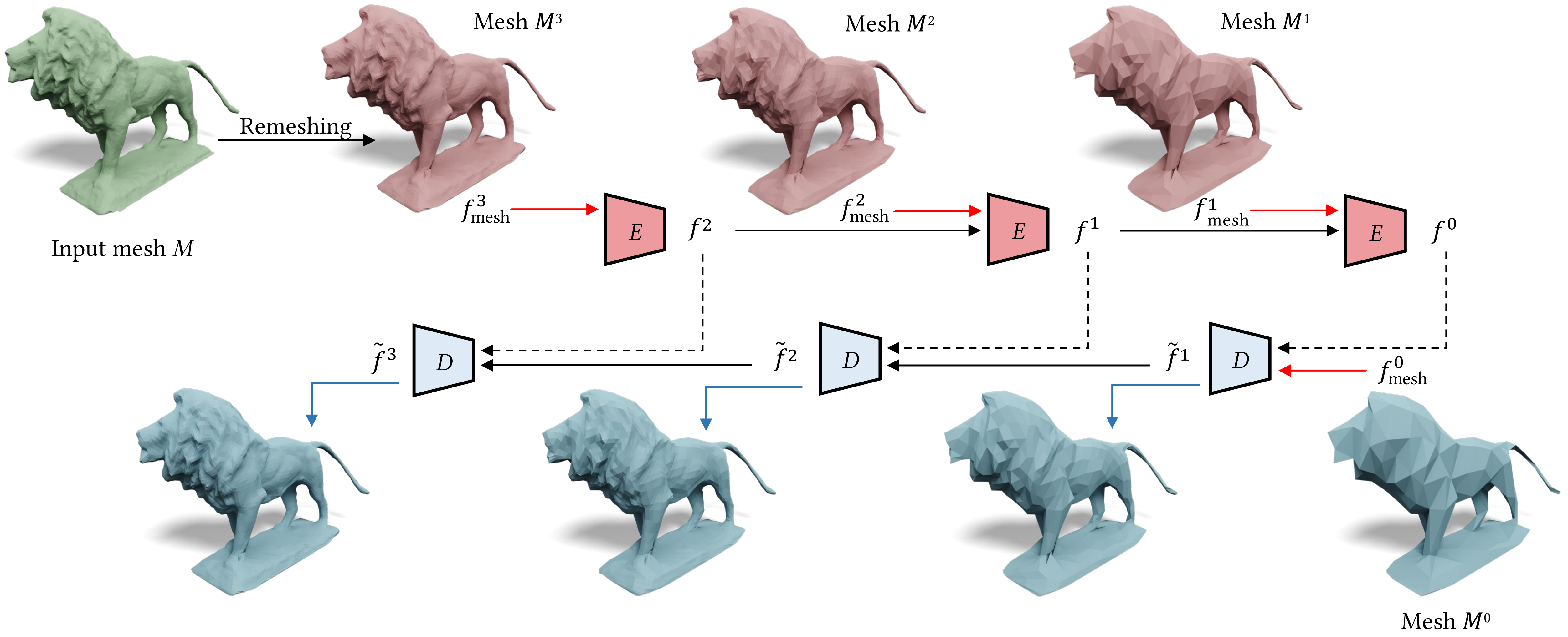
Neural Progressive Meshes



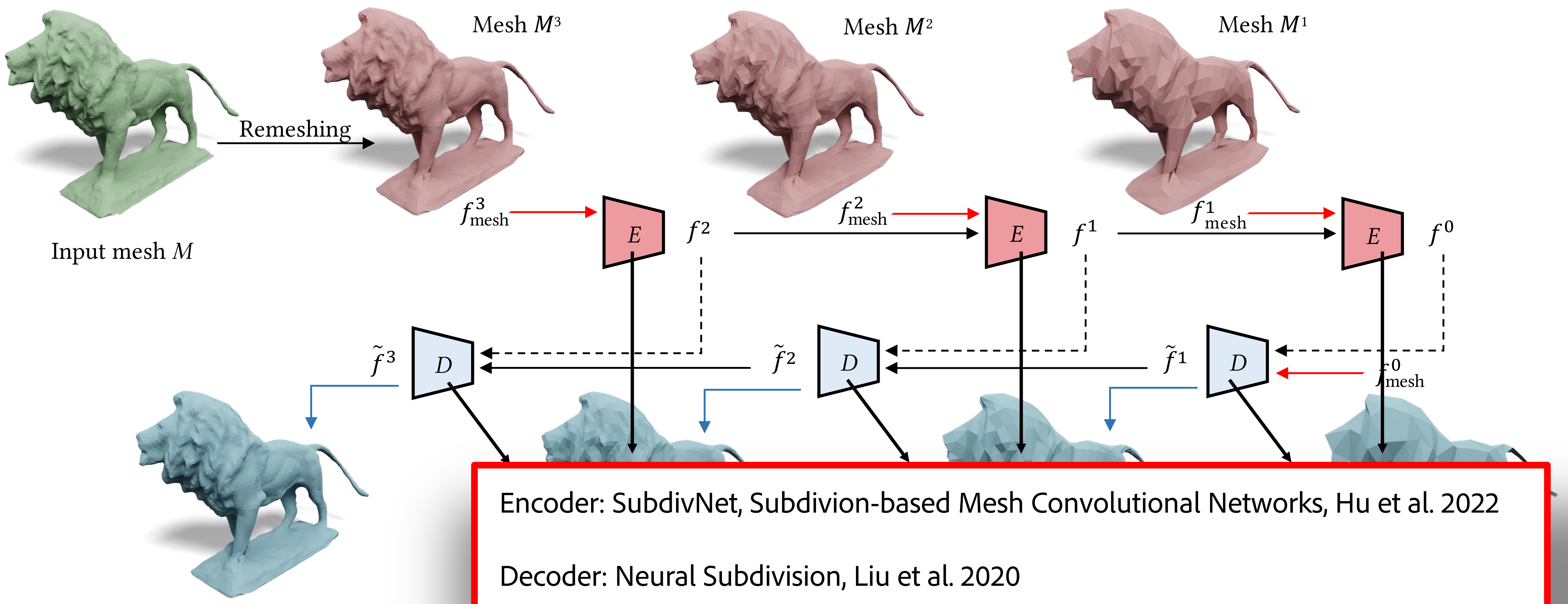
Neural Progressive Meshes



Neural Progressive Meshes



Neural Progressive Meshes

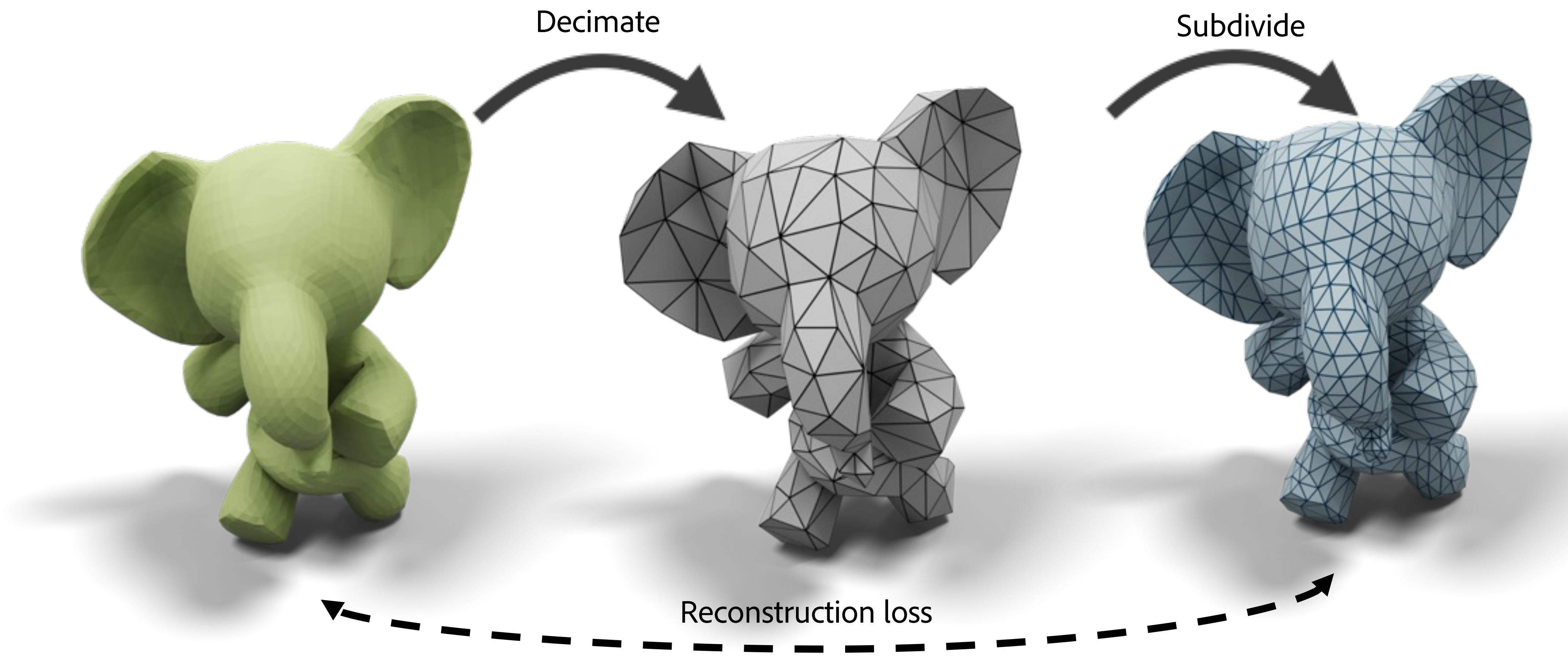


Encoder: SubdivNet, Subdivion-based Mesh Convolutional Networks, Hu et al. 2022

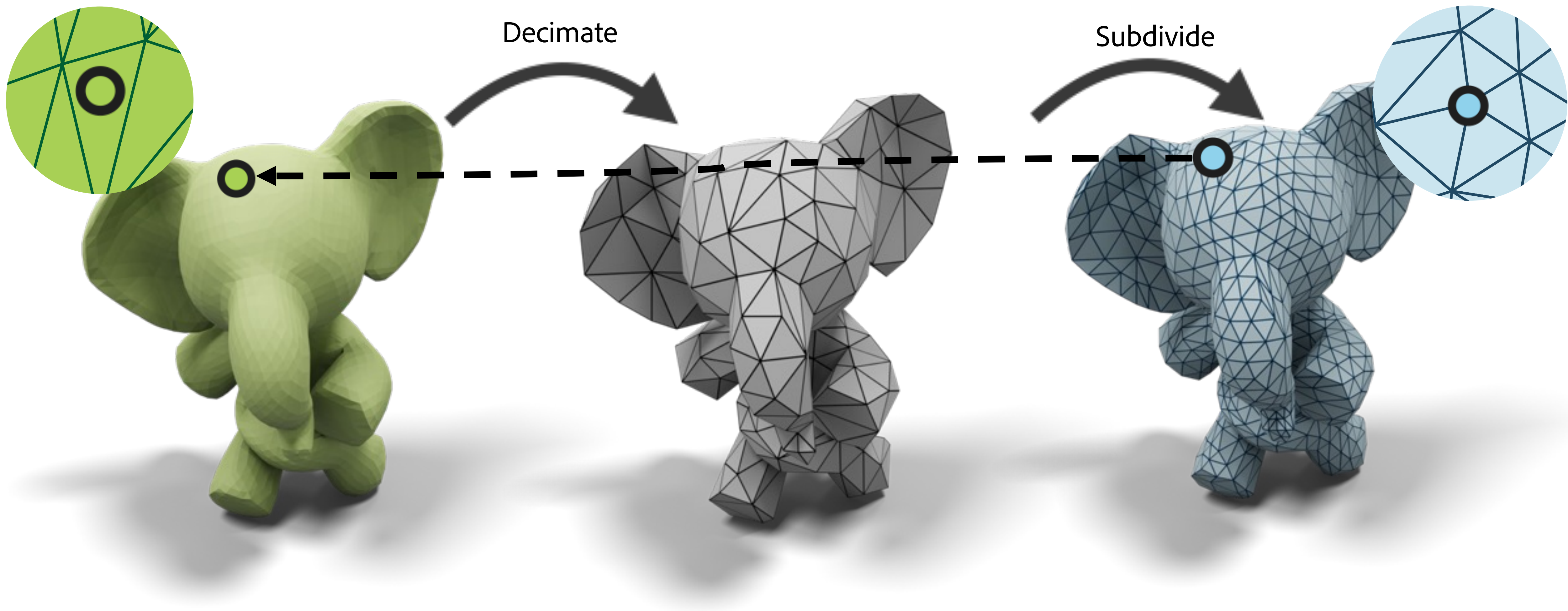
Decoder: Neural Subdivision, Liu et al. 2020

Neural Subdivision

- Decimate high-res mesh to create training data
- Learn local up-sampling filters

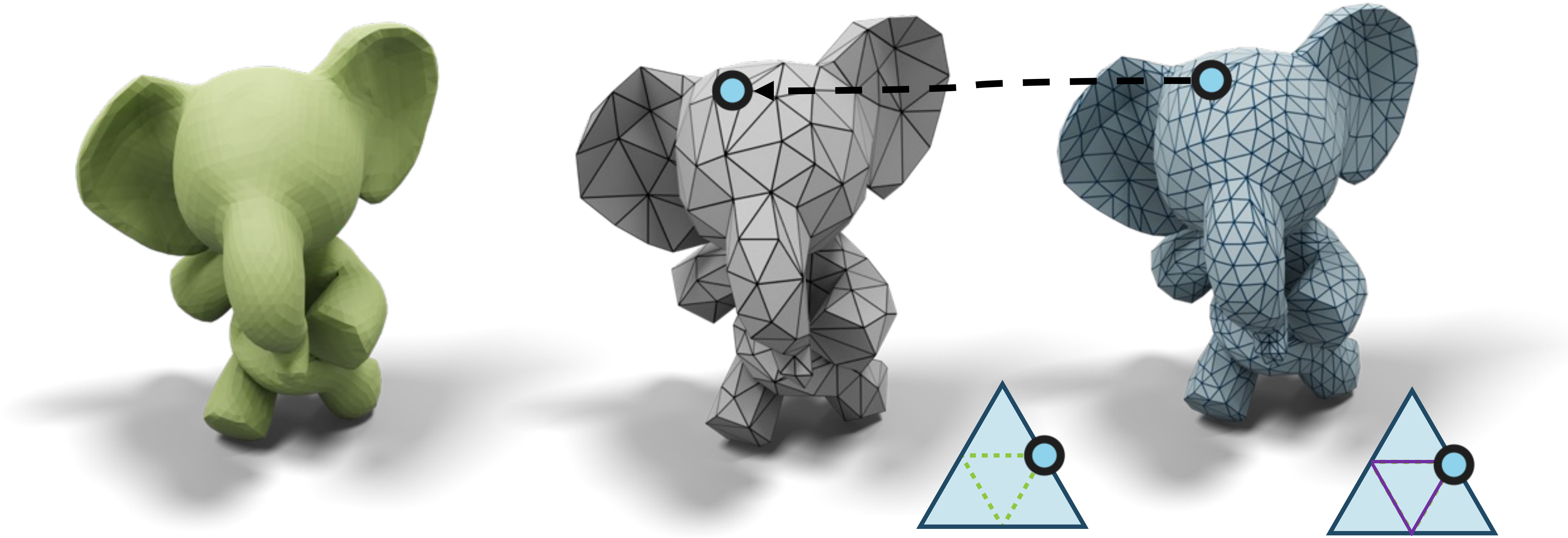


Neural Subdivision: Maintaining Bijective Mapping



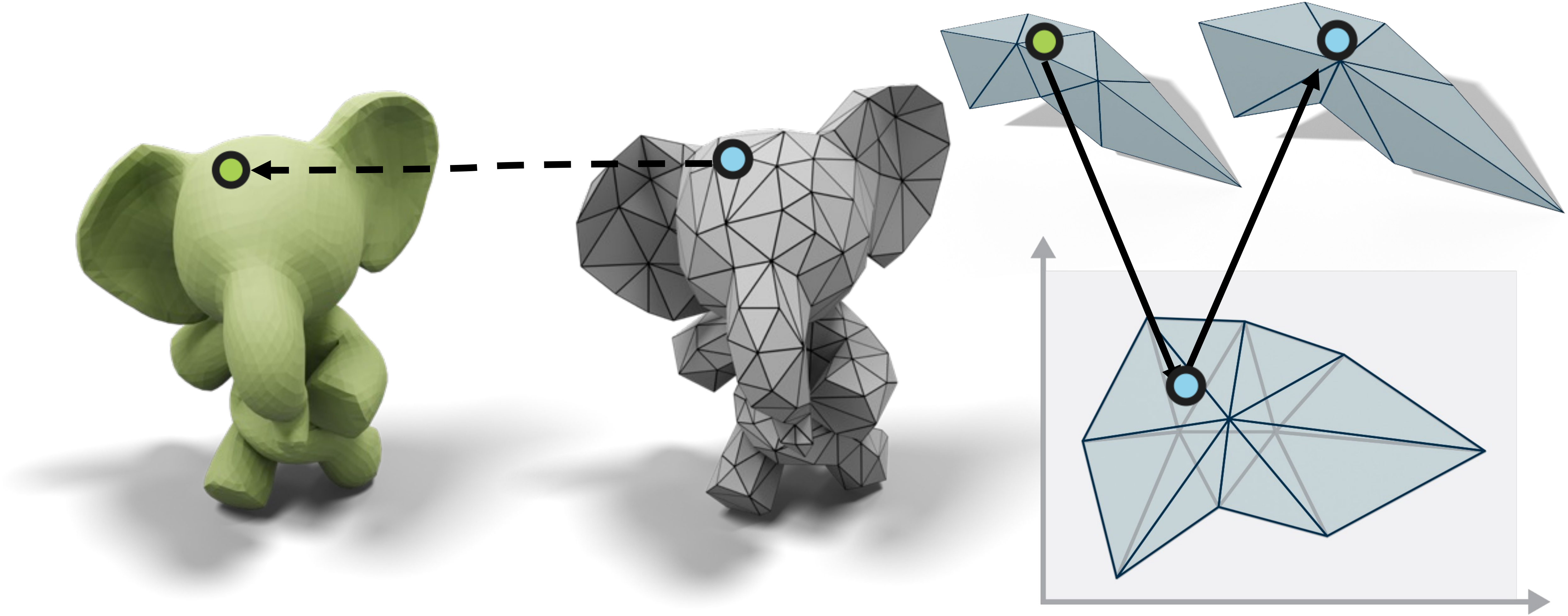
Neural Subdivision: Maintaining Bijective Mapping

- Record barycentric coordinates during subdivision

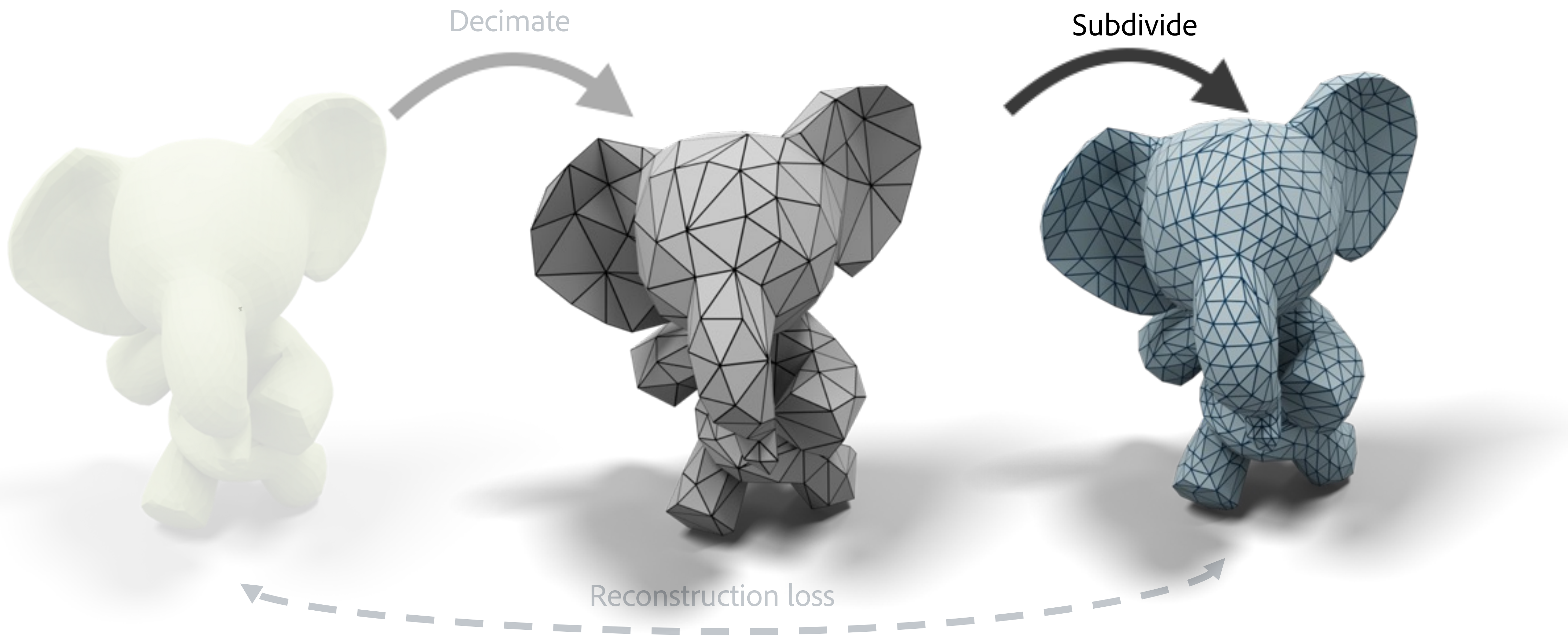


Neural Subdivision: Maintaining Bijective Mapping

- Record barycentric coordinates during subdivision
- Match via parameterization during decimation

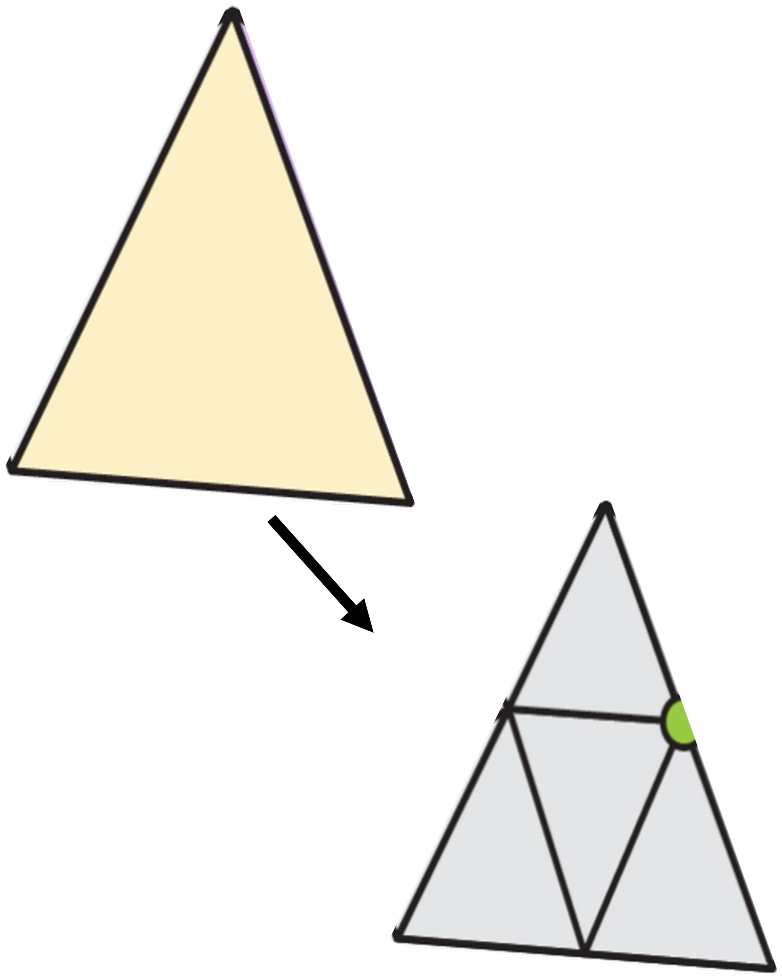


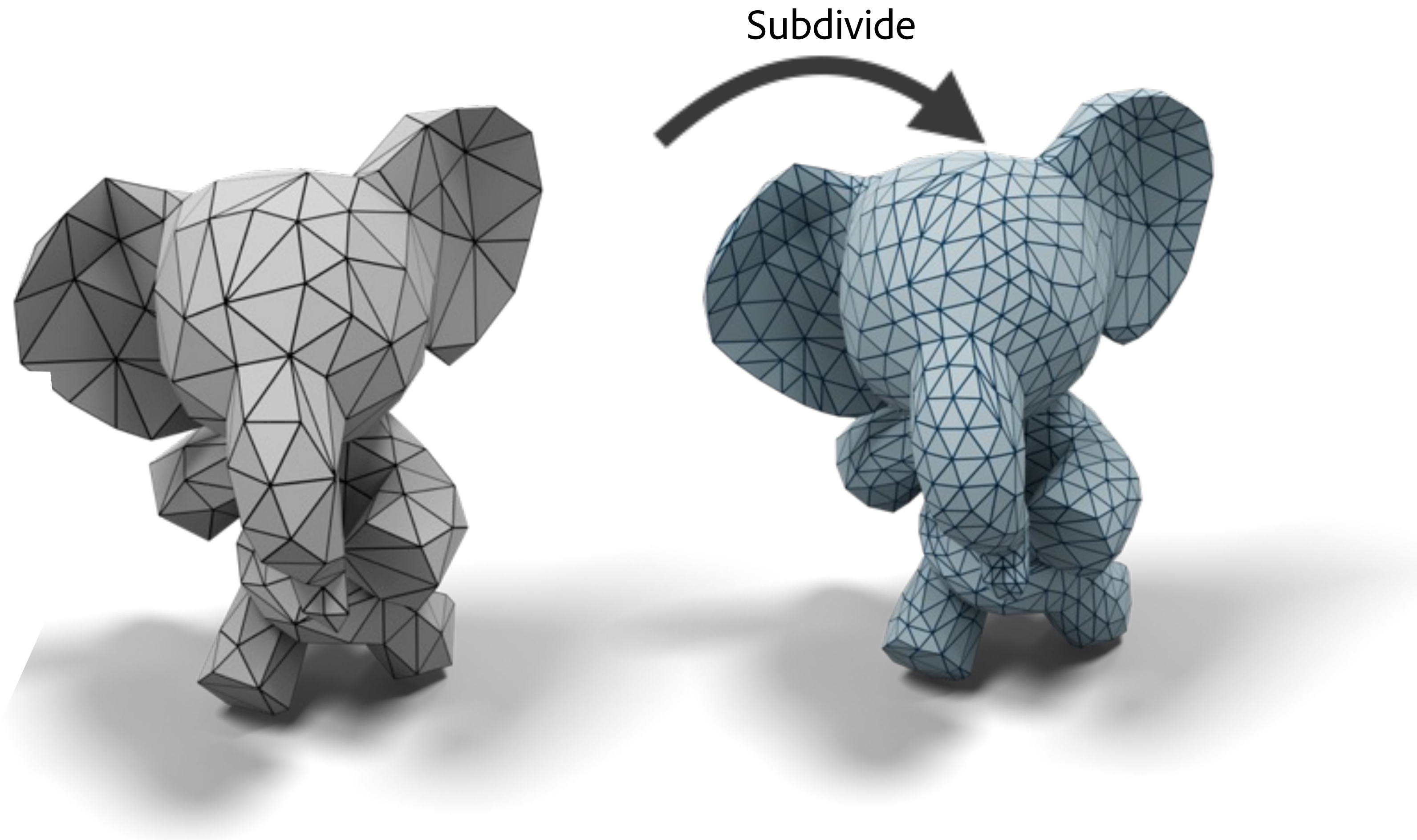
Neural Subdivision



Neural Subdivision

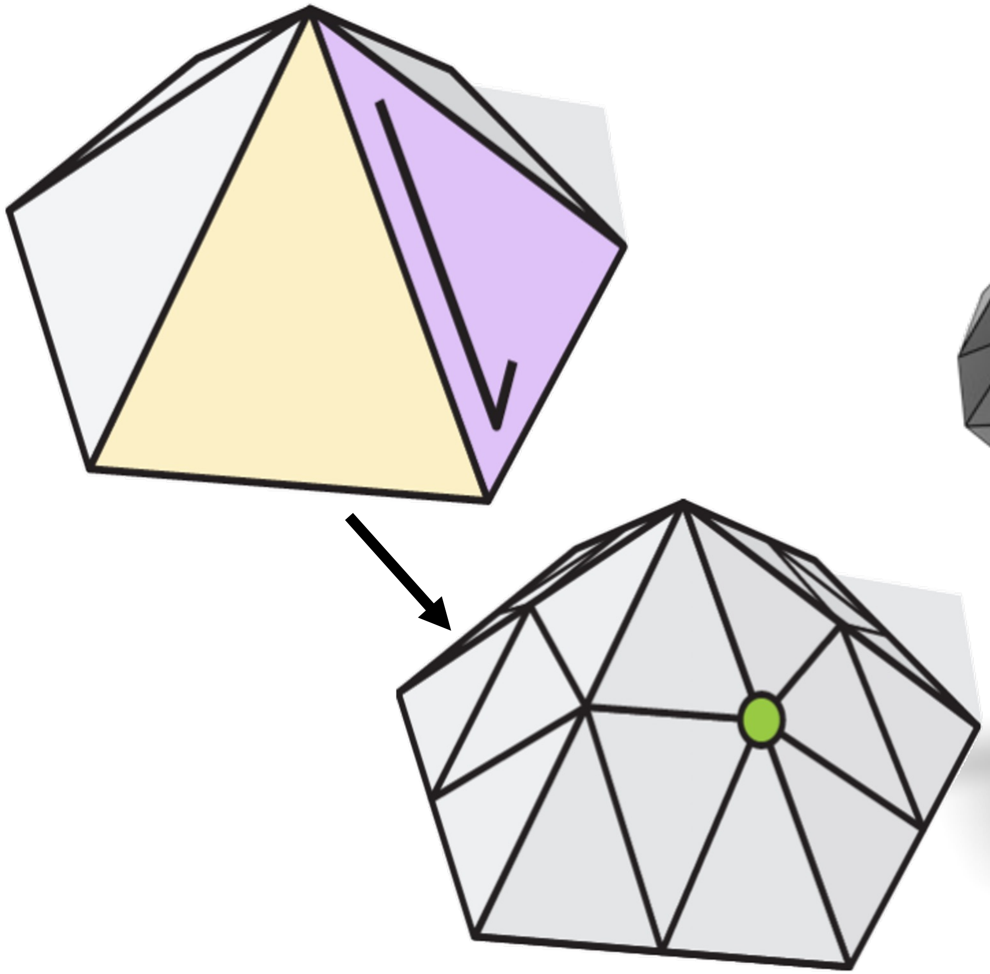
- Triangle Split (mid-edge)

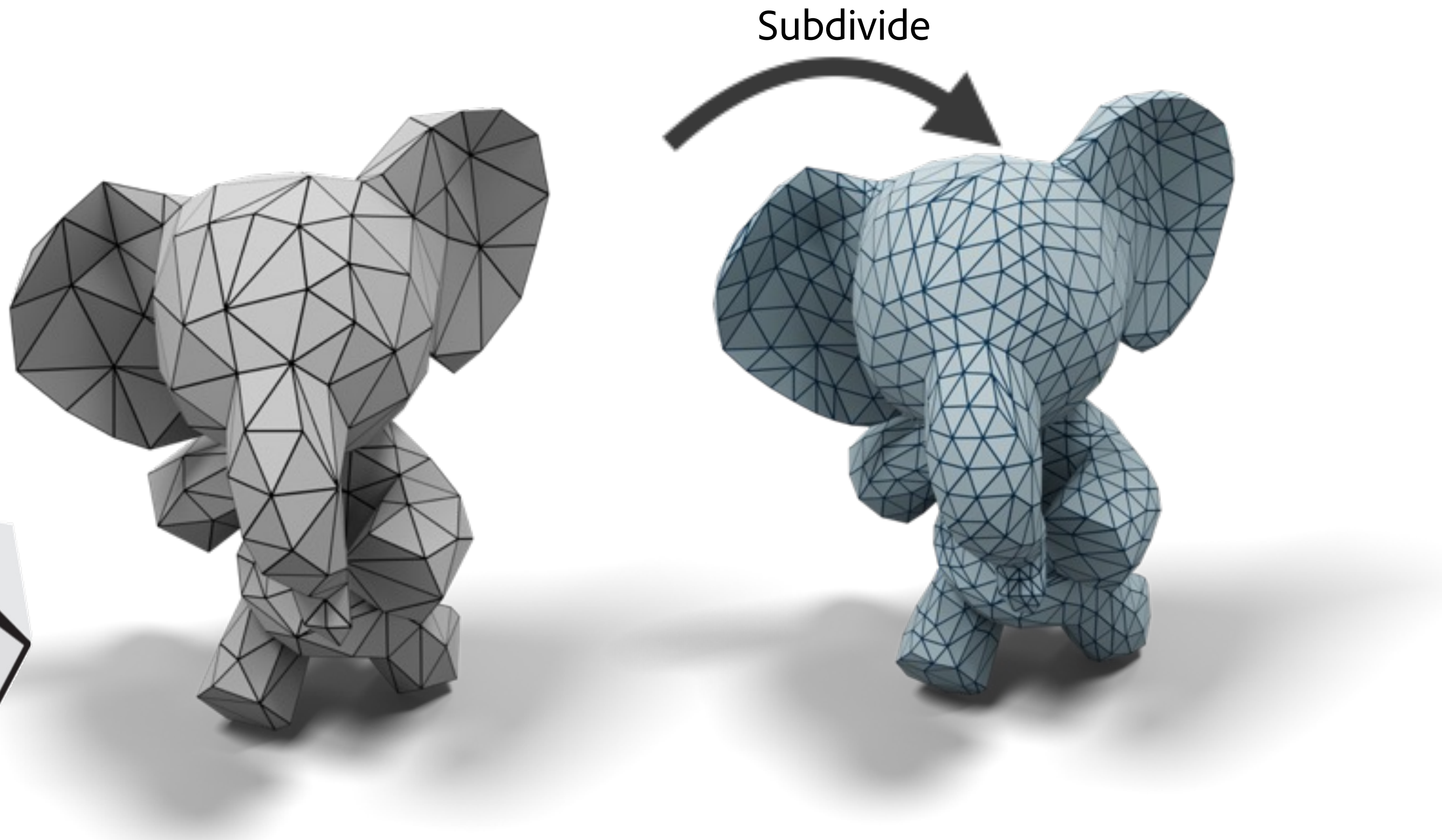




Neural Subdivision

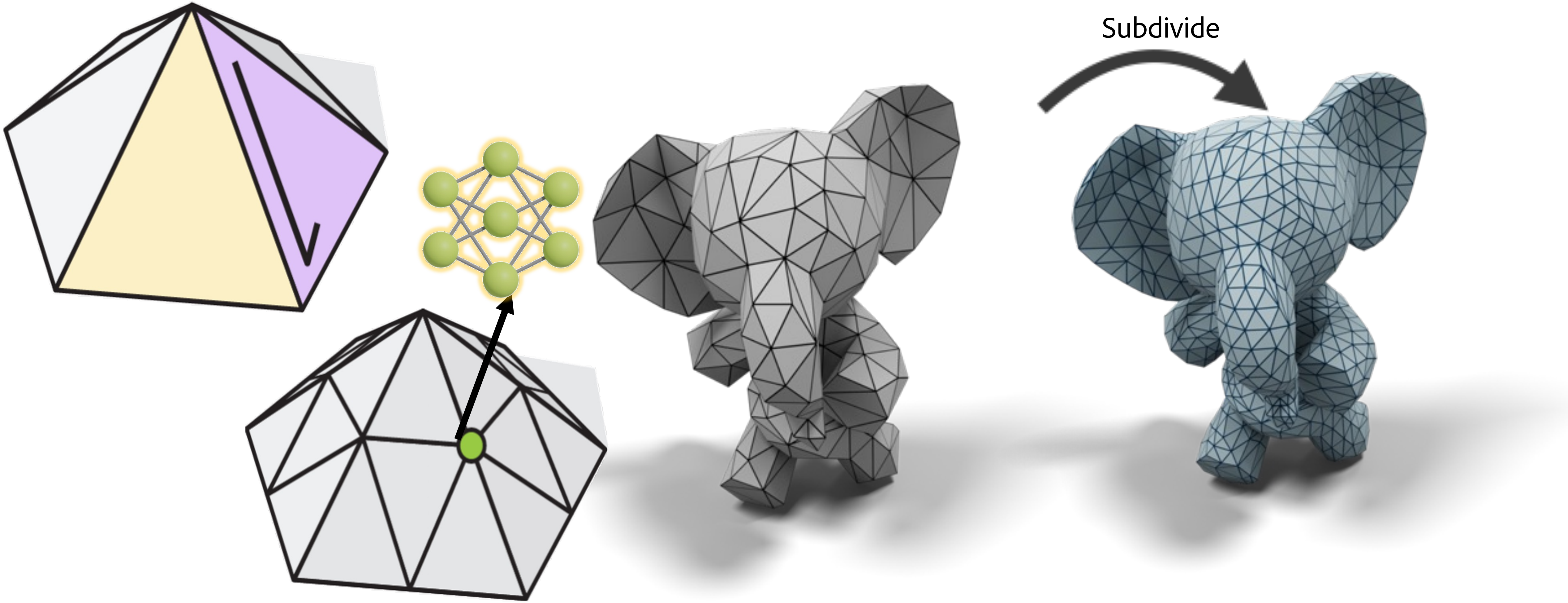
- Triangle Split (mid-edge)





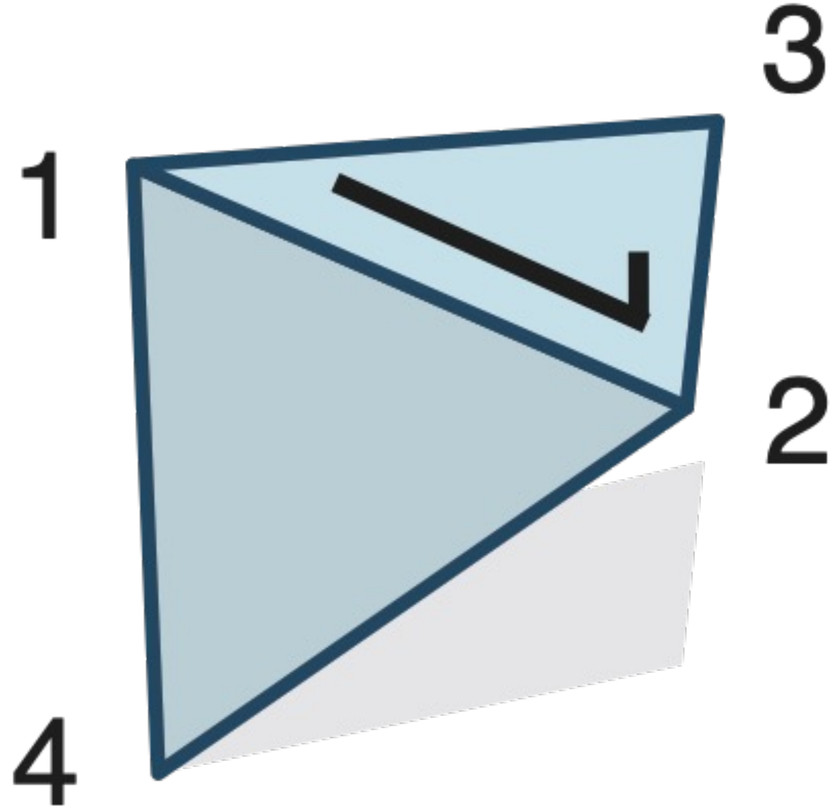
Neural Subdivision

- Triangle Split (mid-edge)
- Set vertex positions via neural network



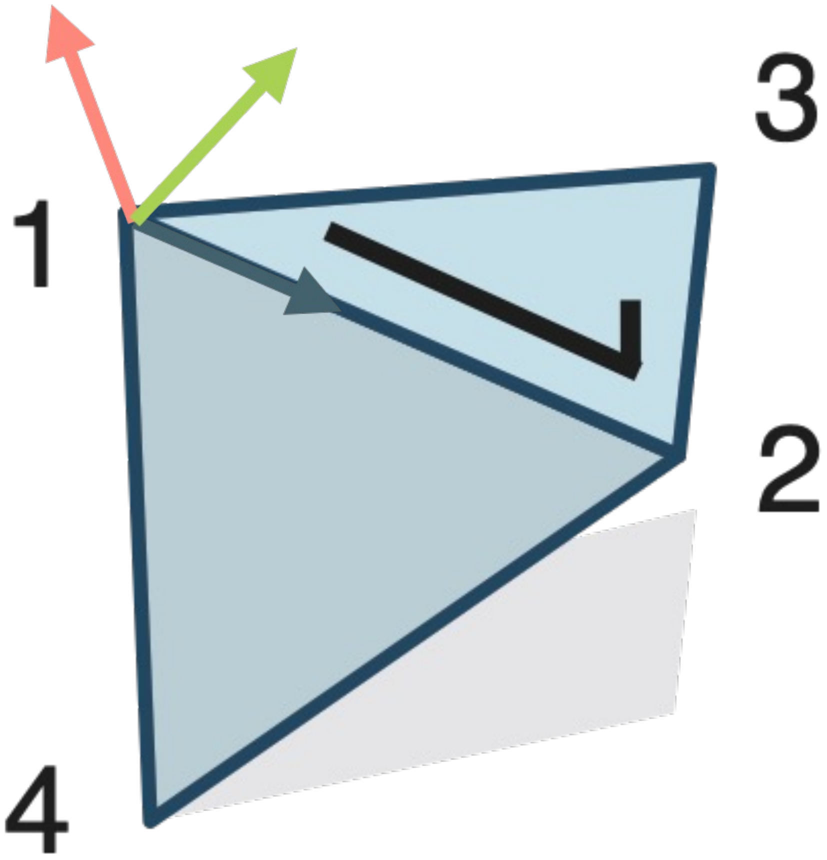
Neural Subdivision: Architecture

- Half-flap: directed edges and two adjacent triangles
  - Fixed Dimensions
  - Canonical Ordering



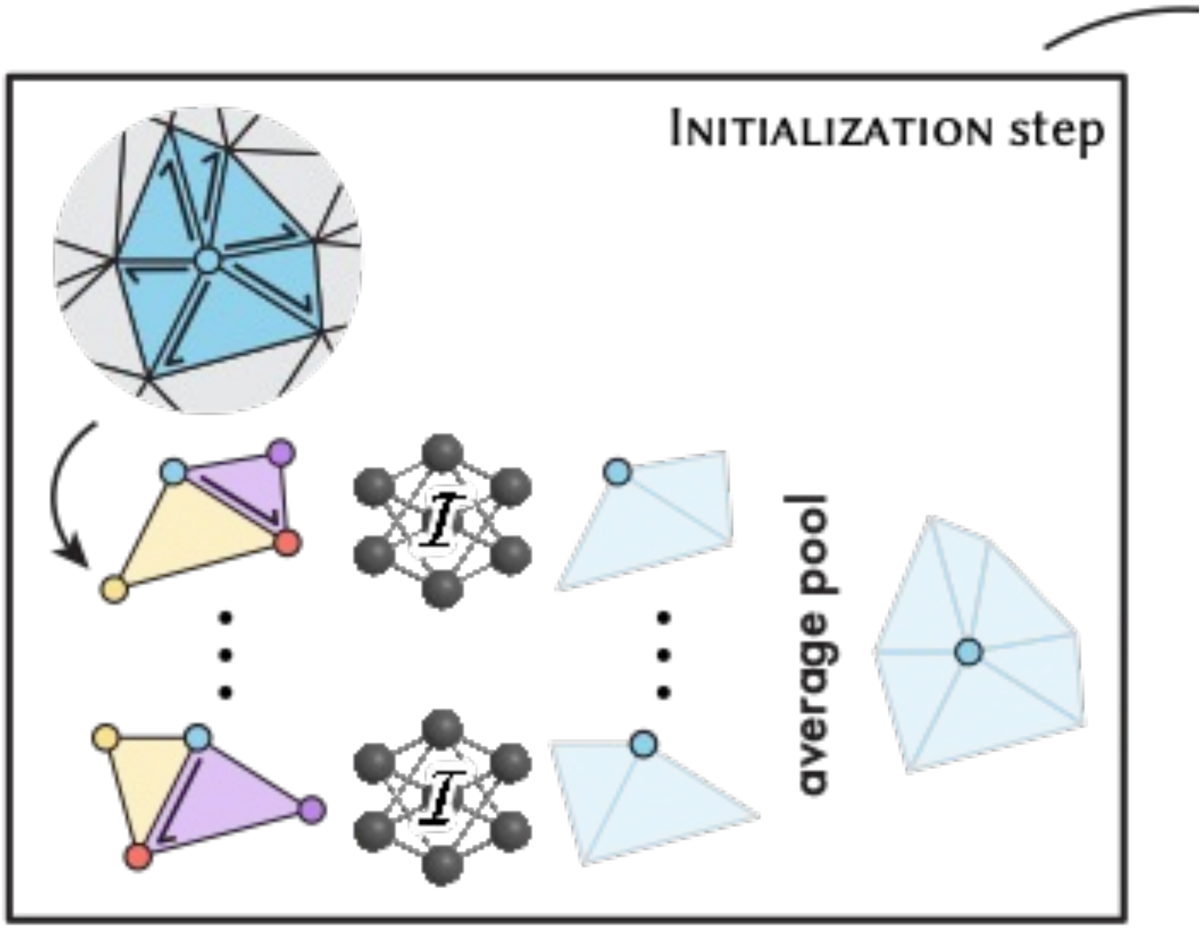
Neural Subdivision: Architecture

- Half-flap: directed edges and two adjacent triangles
- Represent (differential) geometry in flap's local coordinate frame



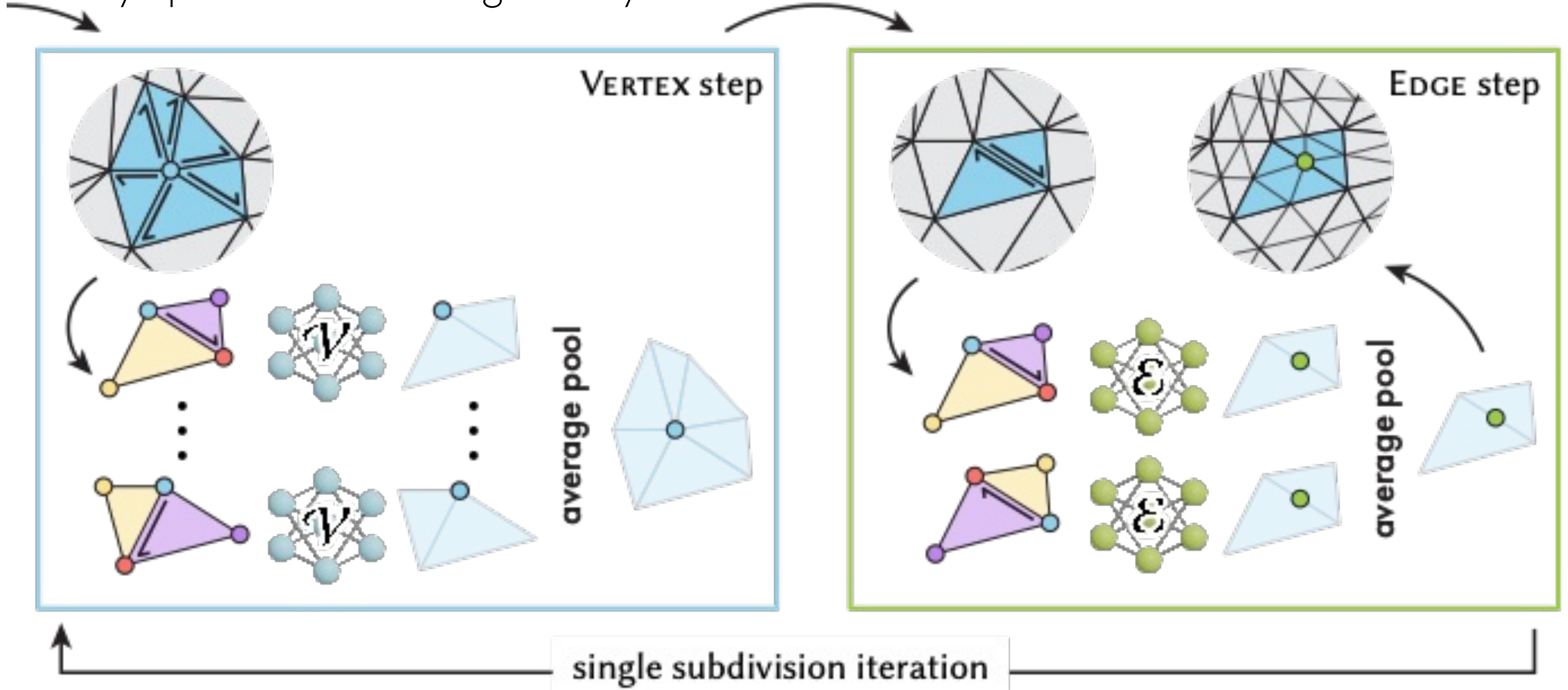
Neural Subdivision: Pipeline

- Initialize per-vertex features



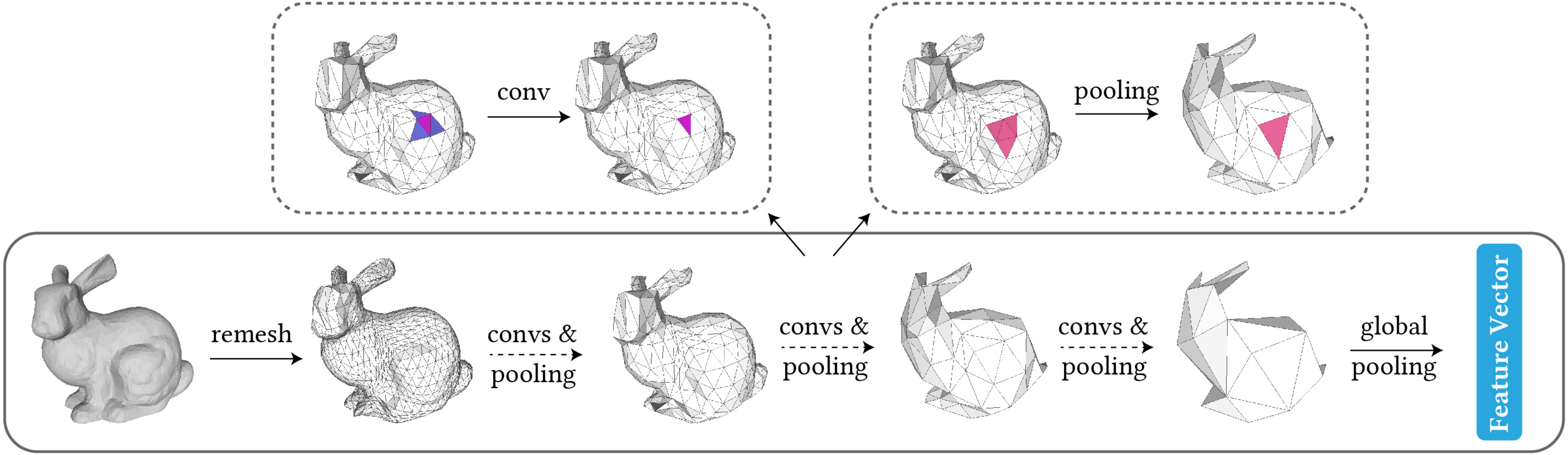
Neural Subdivision: Pipeline

- Initialize per-vertex features
- Iteratively update features and geometry at old & new vertices

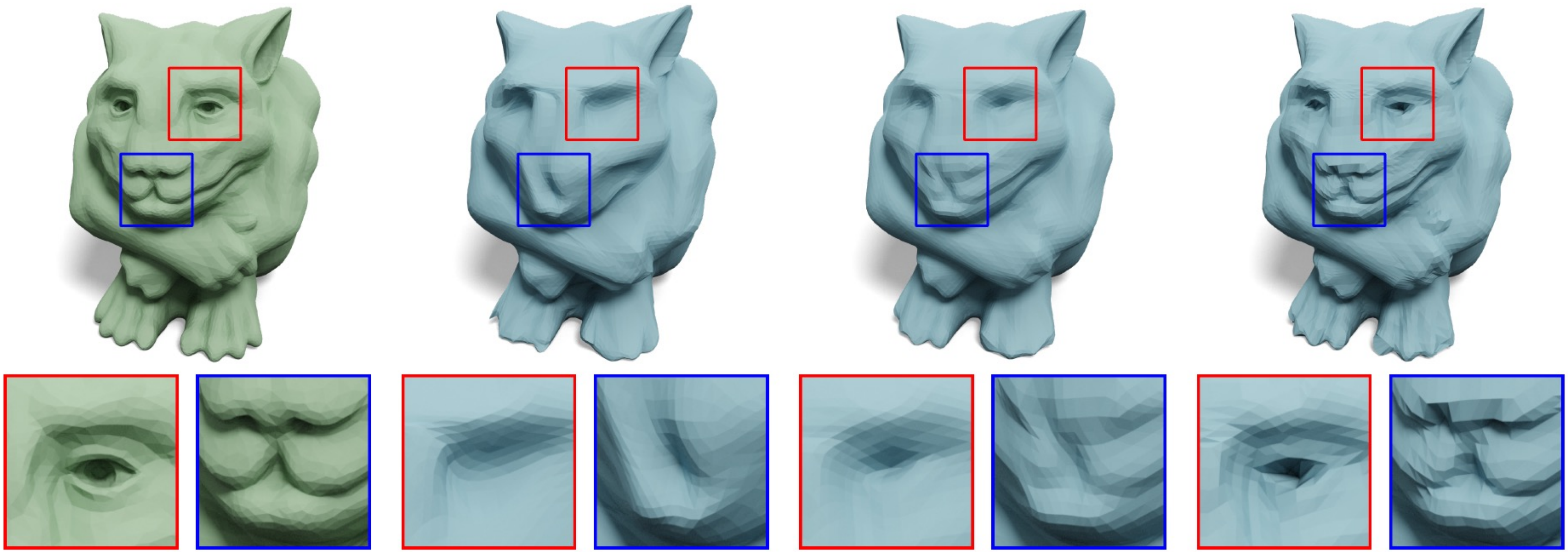


Neural Subdivision: SubdivNet for analysis

- A follow-up work by Hu et al. 2022 showed that Subdivision can also be used for analysis



Neural Progressive Meshes



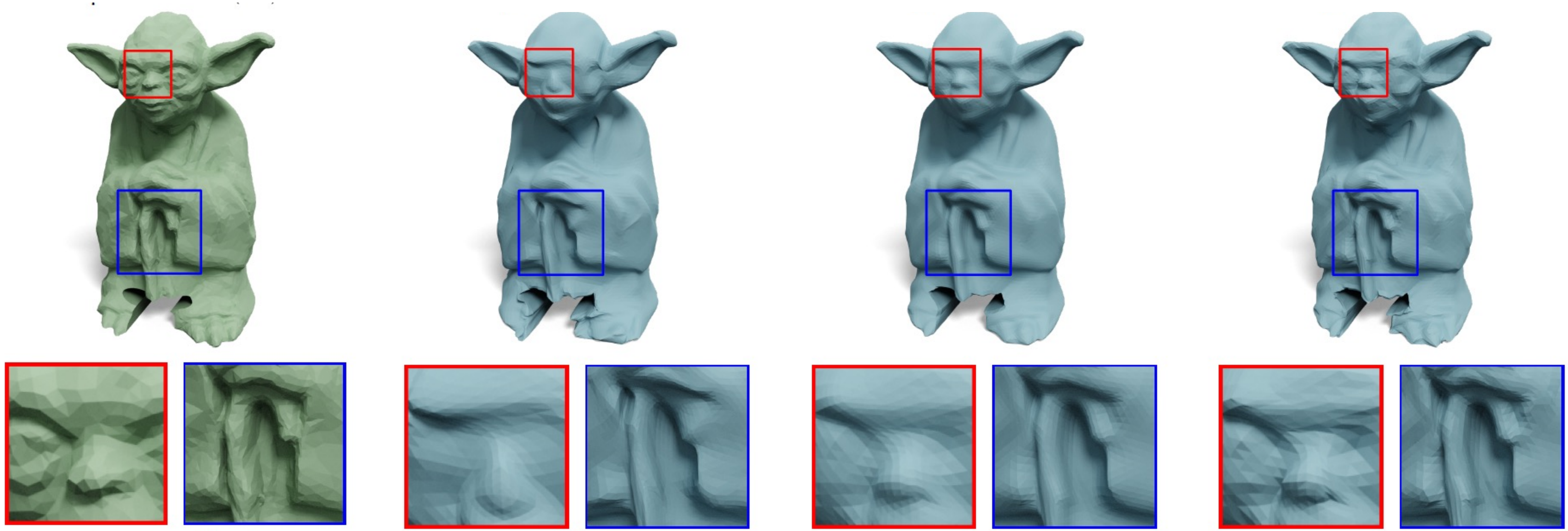
CR / d_{pm} ($\times 10^{-4}$) / d_{normal} — Ground truth

64.73 / 14.87 / 10.33° — Ours w/o features

17.78 / 11.48 / 6.94° — Ours + 40 features

10.31 / 5.37 / 5.31° — Ours + 400 features

Neural Progressive Meshes



CR / d_{pm} ($\times 10^{-4}$) / d_{normal}	39.83 / 15.59 / 11.54°	7.18 / 7.70 / 7.66°	3.17 / 4.89 / 8.15°
Ground truth	Ours w/o features	Ours + 40 features	Ours + 400 features

Neural Progressive Meshes Takeaways

Use classical Geometry Processing modules as layers in NN, e.g.:

- Subdivision
- Decimation

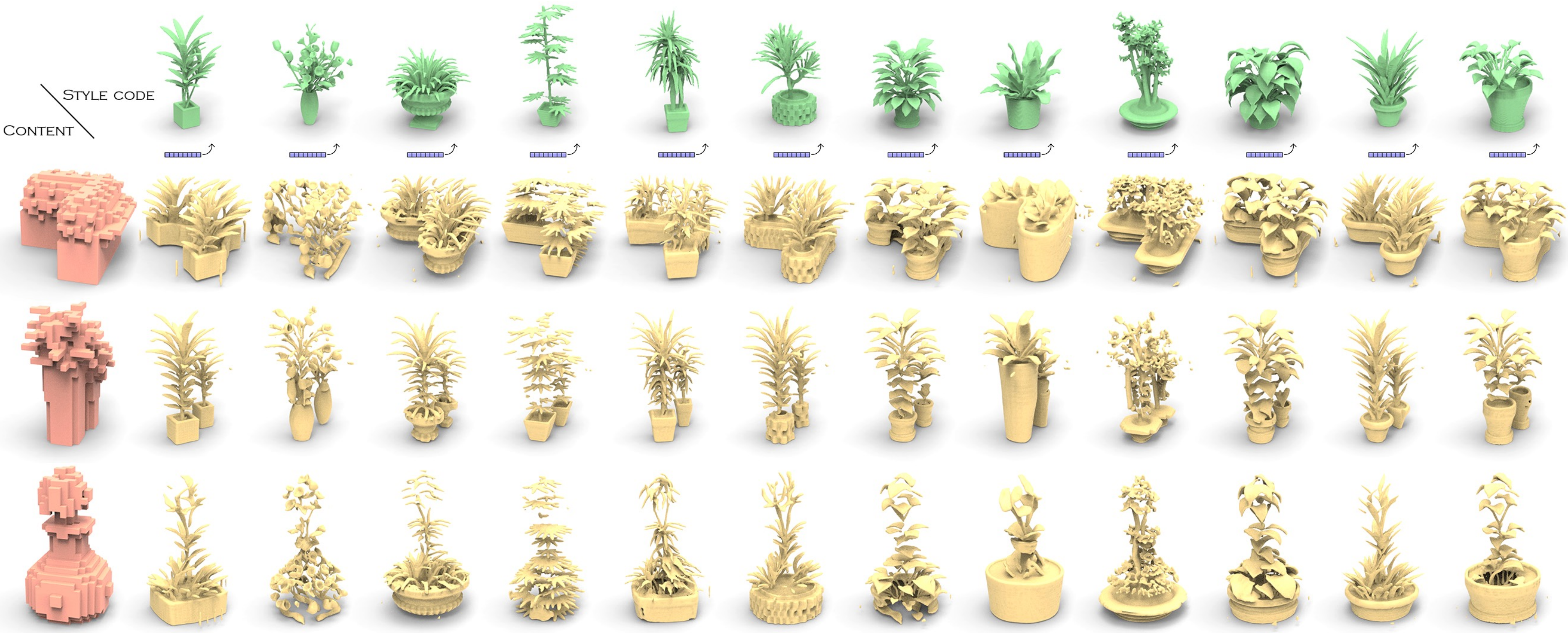
Neural Networks can:

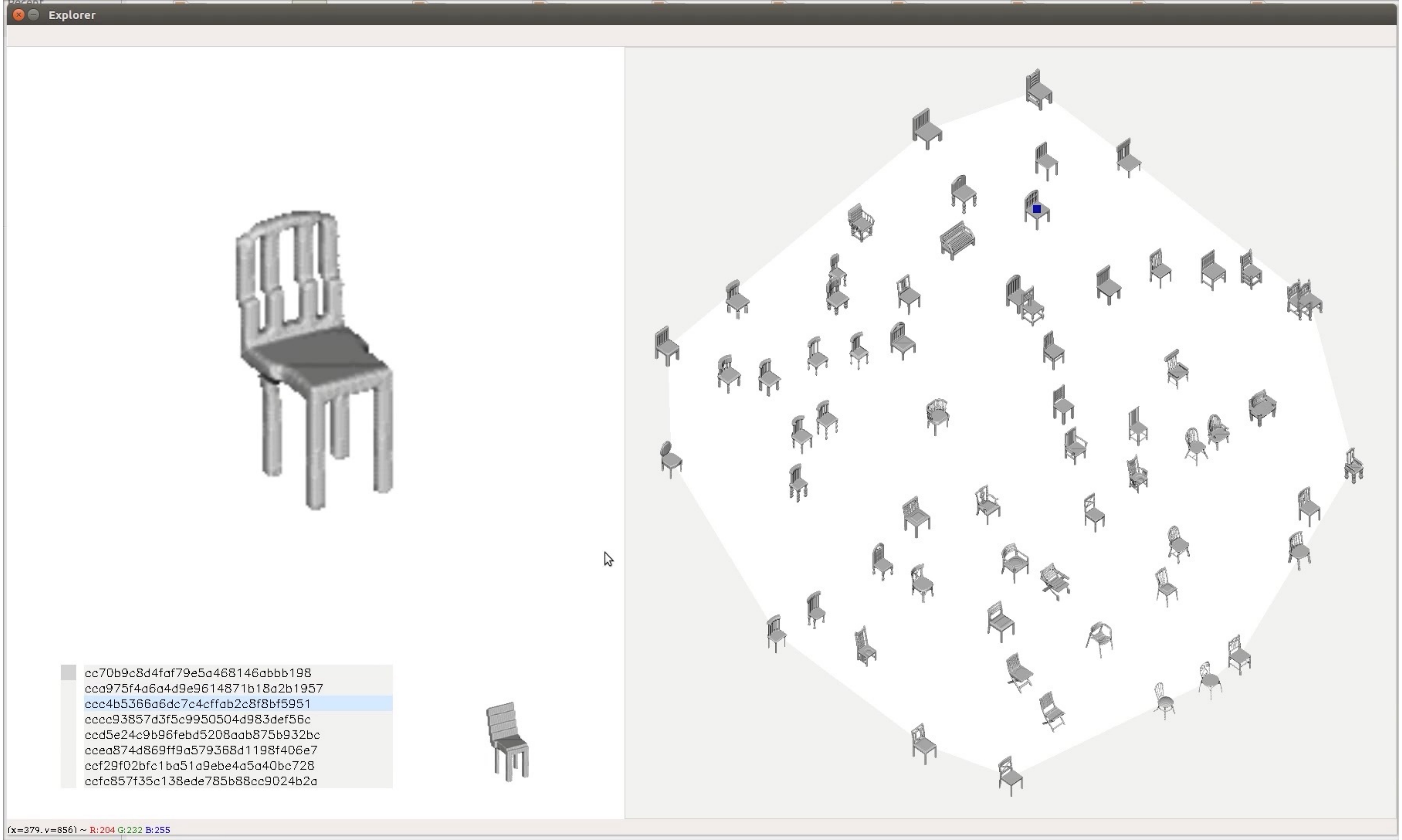
- Implicitly learn relations between shapes during training

Questions for the Future Work

- How to leverage pre-trained visual networks to get prior on local geometric details?
- Can we use subdivision for Neural Detailization?

Example of a Neural Detailization Method: Décor-GAN




Explorer
cc70b9c8d4faf79e5a468146abbb198
cca975f4a6a4d9e9614871b18a2b1957
ccc4b5366a6dc7c4cffab2c8f8bf5951
cccc93857d3f5c9950504d983def56c
ccd5e24c9b96febd5208aab875b932bc
ccea874d869ff9a579368d1198f406e7
ccf29f02bfc1ba51a9ebe4a5a40bc728
ccfc857f35c138ede785b88cc9024b2a
(x=379, y=856) ~ R:204 G:232 B:255

Future Work

Leverage additional priors in Neural Geometry Modeling and Analysis

- Large language models
- Large language and image co-embedding models
- Large generative models for images

Leverage task-specific geometry processing tools in designing architectures

- Differentiable layers
- Task-specific loss functions and regularization terms
- Rigorous representations

Support real workflows used by artists and designers

- HCI will be at the core of any innovation
- Our tools should not compete with people

Collaborators

- Project Leads
  - Yifan Wang, ETH Zurich (Neural Cages for Detail-Preserving 3D Deformations, **CVPR 2020 oral**)
  - Noam Aigerman, Adobe (Neural Jacobian Fields: Learning Intrinsic Mappings of Arbitrary Meshes, **SIGGRAPH 2022**)
  - William Gao, U. of Chicago (TextDeformer: Geometry Manipulation using Text Guidance, **SIGGRAPH 2023**)
  - Richard Liu, U. of Chicago (DA Wand: Distortion-Aware Selection using Neural Mesh Parameterization, **CVPR 2023**)
  - Yun-Chun Chen, U. of Toronto (Neural Progressive Meshes, **SIGGRAPH 2023**)
  - Hsueh-Ti (Derek) Liu, U. f Toronto (Neural Subdivision, **SIGGRAPH 2020**)
  - Zhiqin Chen, Simon Fraser University (DECOR-GAN: 3D Shape Detailization by Conditional Refinement, **CVPR 2021 oral**)

- Collaborators
  - Siddhartha Chaudhuri, Thibault Groueix, Jun Saito, Alec Jacobson– **Adobe Research**
  - Rana Hanocka – **U. of Chicago**
  - Olga Sorkine – **ETH Zurich**
  - Richard Zhang **– Simon Fraser University**
  - Kunal Gupta – **UCSD**