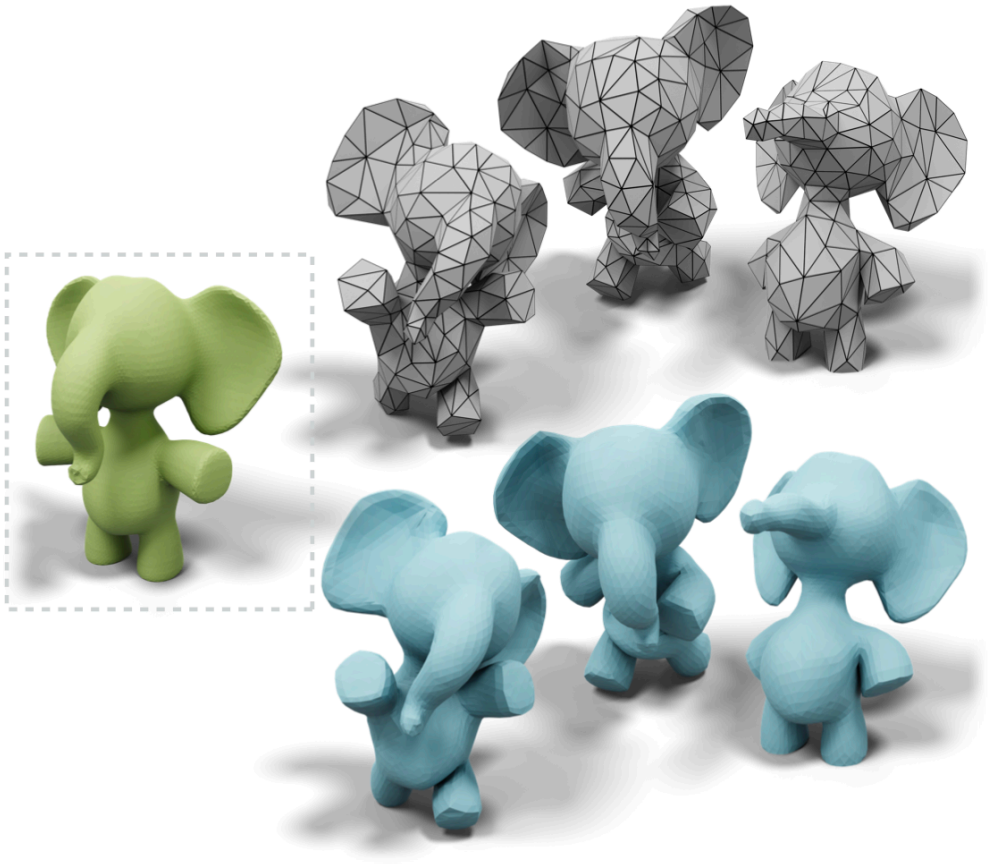
Neural Subdivision

Table of Content

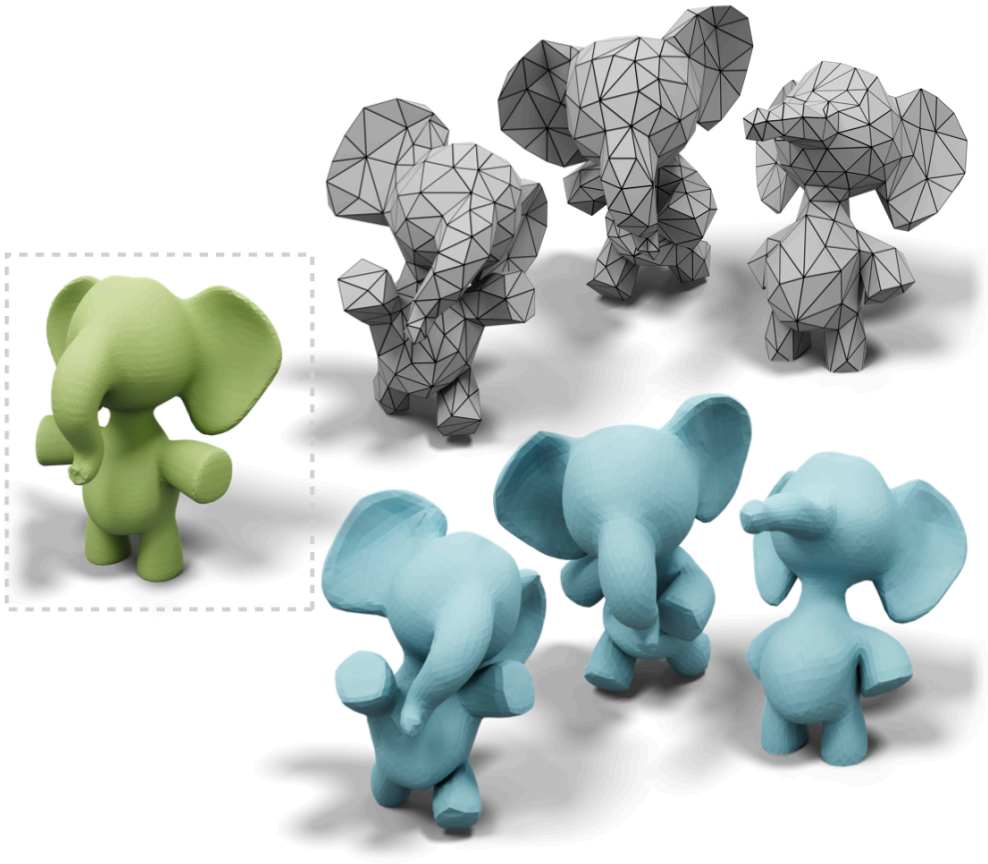
What is Subdivision?



【Application】

- Interactive modeling
  A modeler may start with a very coarse cage, adjust vertex positions, then subdivide once, adjust the finer mesh vertices, and repeat this process until satisfied
- Render for points of interest

What is Subdivision?



【Classic Methods】

Core idea: recursive up-sampling of a discrete surface mesh

- **divide input mesh into vertices**
  - split edges
  - add vertices
- **linear weighted average**: smooth positions of the mesh vertices

What is Subdivision?

【Classic Methods】

Core idea: recursive up-sampling of a discrete surface mesh

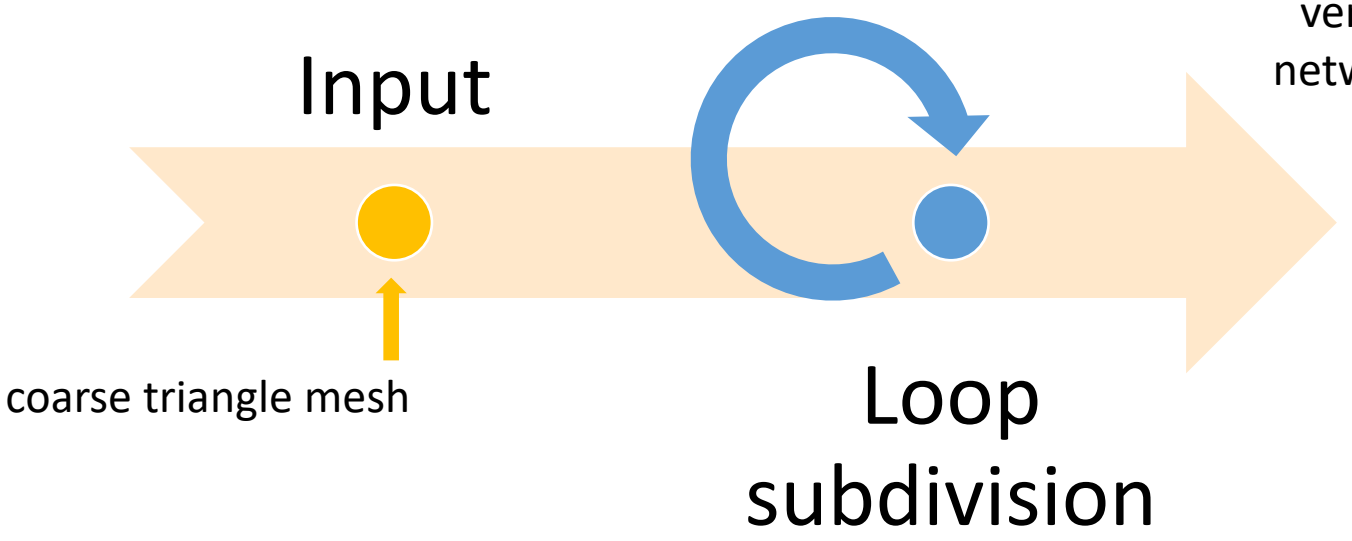
- **divide input mesh into vertices**
  - split edges
  - add vertices
- **linear weighted average**: smooth positions of the mesh vertices

【Weakness】

- Usually overly smooth the entire shape
- Details loss
- Fixed on-size-fits-all weighting rules globally(Unuse lots of information)

What is Neural Subdivision?

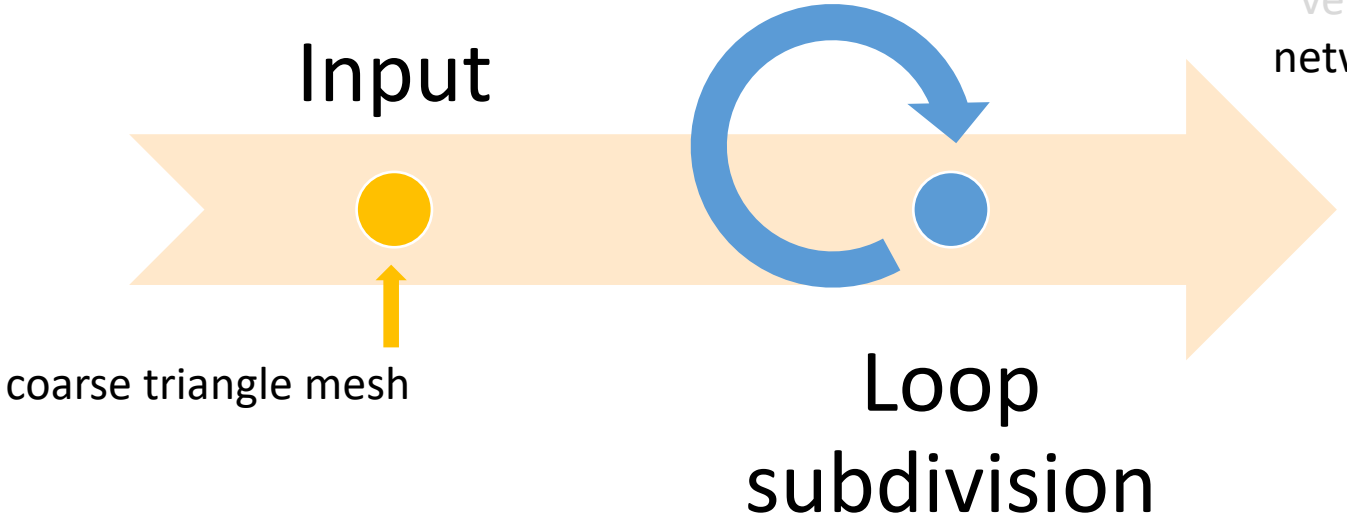
A novel framework for data-driven coarse-to-fine geometry modeling.



recursively subdivide and predicting vertex positions using a neural network conditioned on the local geometry of a patch

What is Neural Subdivision?

A novel framework for data-driven coarse-to-fine geometry modeling.



recursively subdivide and predicting vertex positions using a neural network conditioned on the local geometry of a patch

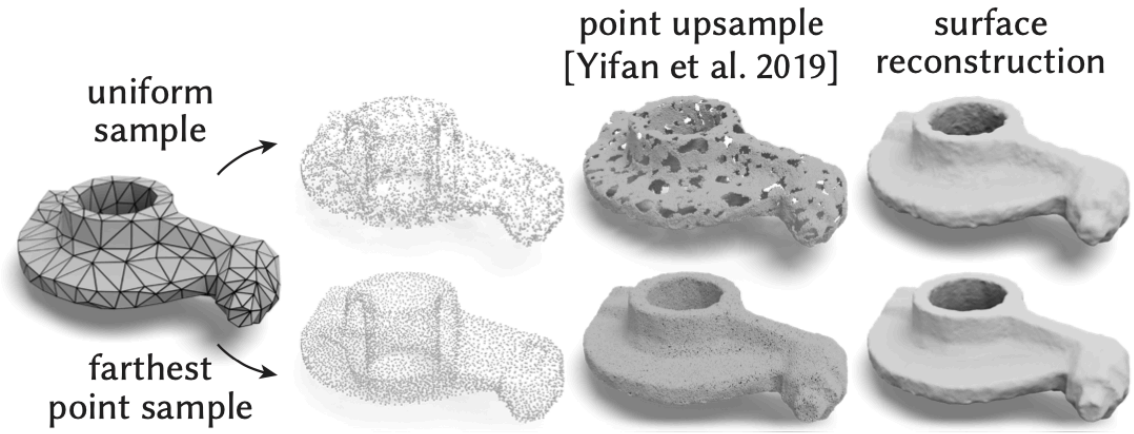
?

Training data pairs

Neural Geometry Learning

【Deep point cloud upsampling】

- lack connectivity information
- require the neural network to estimate the structure of the underlying manifold
- post process is often required to convert the output of point-based methods to meshes(not end-to-end trainable system)



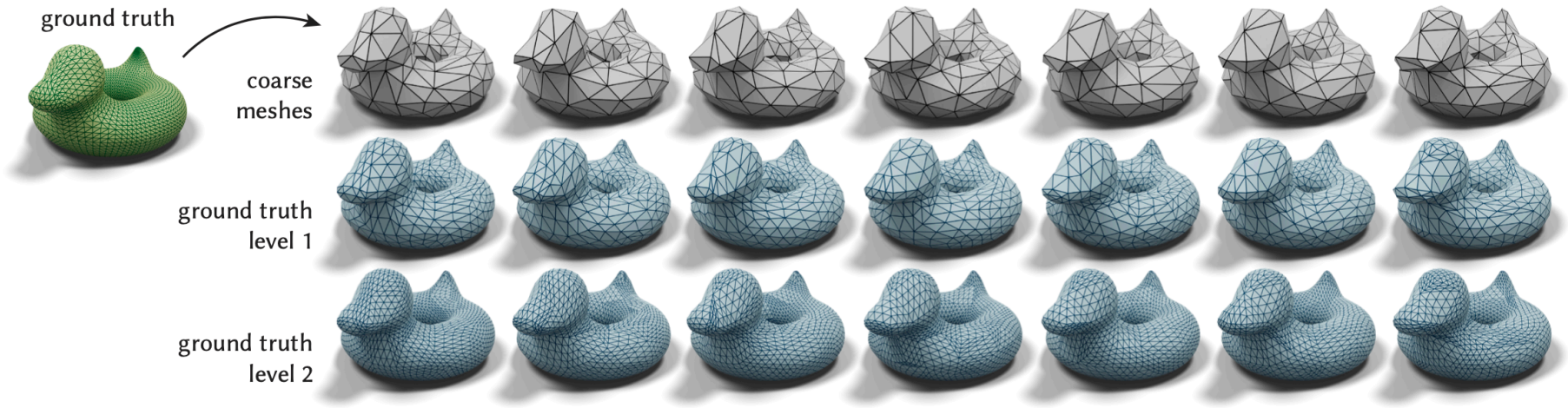
【Others】

- deform global template
- using a local or global parameterization to unfold a mesh into 2D grid
- apply graph-based techniques adapted for mesh

Neural Geometry Learning

【Paper Approach】

Randomly generate low-resolution versions of training samples while maintaining bijection between their surfaces.



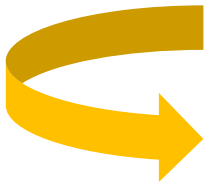
Neural Geometry Learning

【Paper Approach】

Randomly generate low-resolution versions of training samples while maintaining bijection between their surfaces.

【MeshCNN】

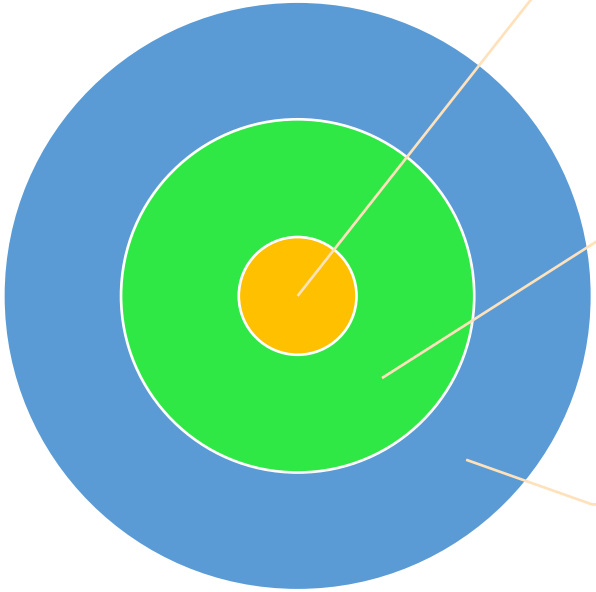
Deterministic tasks -> learn filter over the local mesh structure via undirected edges



Generative tasks -> develop features over the half-flaps

Table of Content

Overview



Training and Loss

Dataset: develop a training data comprising of coarse and fine meshes with bijective mappings between them

l^2 **loss:** distance between each predicted vertex position at every level of subdivision and its corresponding point on the original shape

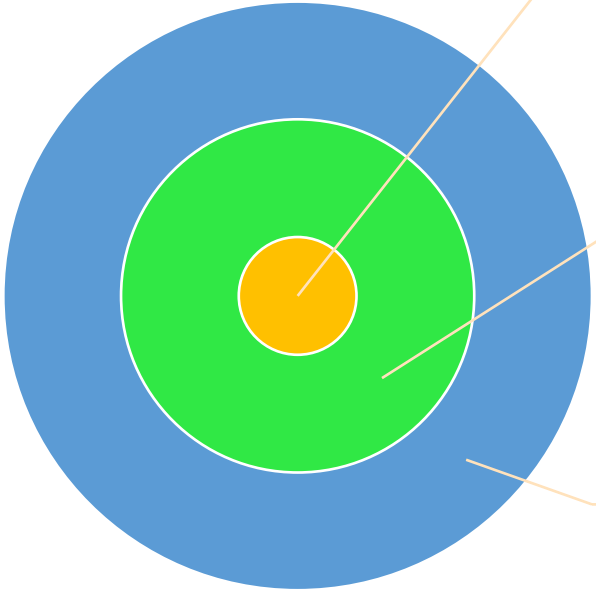
Data Generation

each subdivided mesh at any level can be mapped back to the initial coarse mesh $v\ g\ f$

Network Architecture

- Initialization step
- Vertex step
- Edge step

Overview



Training and Loss

- **Dataset**: develop a training data comprising of coarse and fine meshes with bijective mappings between them
- l^2 **loss:** distance between each predicted vertex position at every level of subdivision and its corresponding point on the original shape

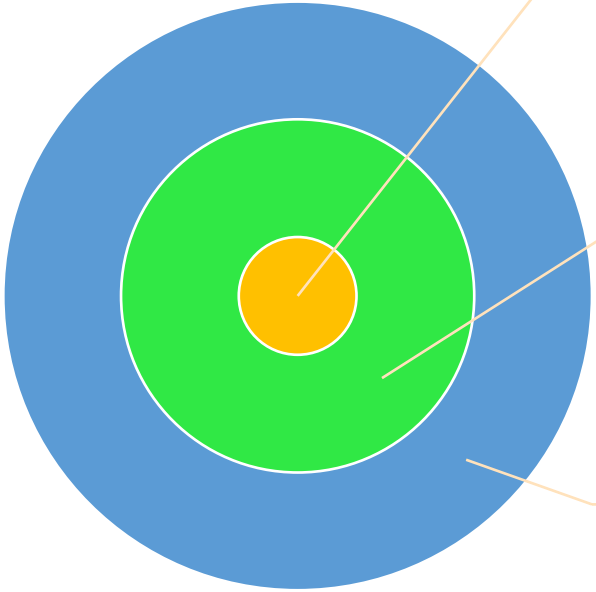
Data Generation

each subdivided mesh at any level can be mapped back to the initial coarse mesh $v\ g\ f$

Network Architecture

- **Initialization step**
- **Vertex step**
- **Edge step**

Overview



Training and Loss

Dataset: develop a training data comprising of coarse and fine meshes with bijective mappings between them

l^2 **loss:** distance between each predicted vertex position at every level of subdivision and its corresponding point on the original shape

Data Generation

each subdivided mesh at any level can be mapped back to the initial coarse mesh $v\ g\ f$

Network Architecture

- **Initialization step**
- **Vertex step**
- **Edge step**

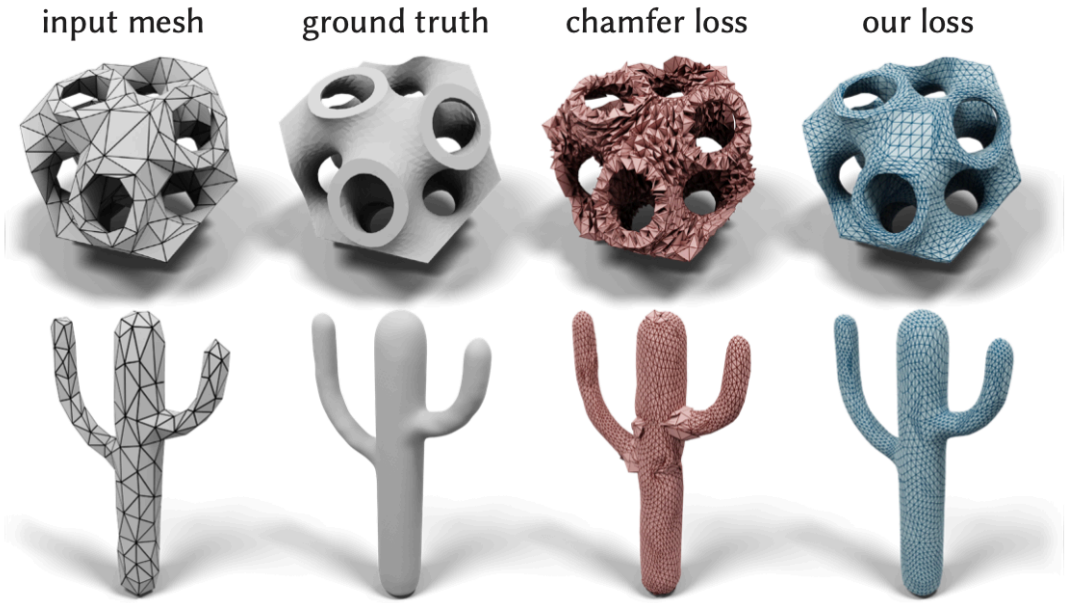
Training and Loss

【Naive Subdivision Training】

- Random sampling pairs of coarse/fine meshes
- measure the distance between the network's predicted subdivision and the ground truth
- iterate over coarse/fine pairs while optimizing the loss

【Chamfre Loss】

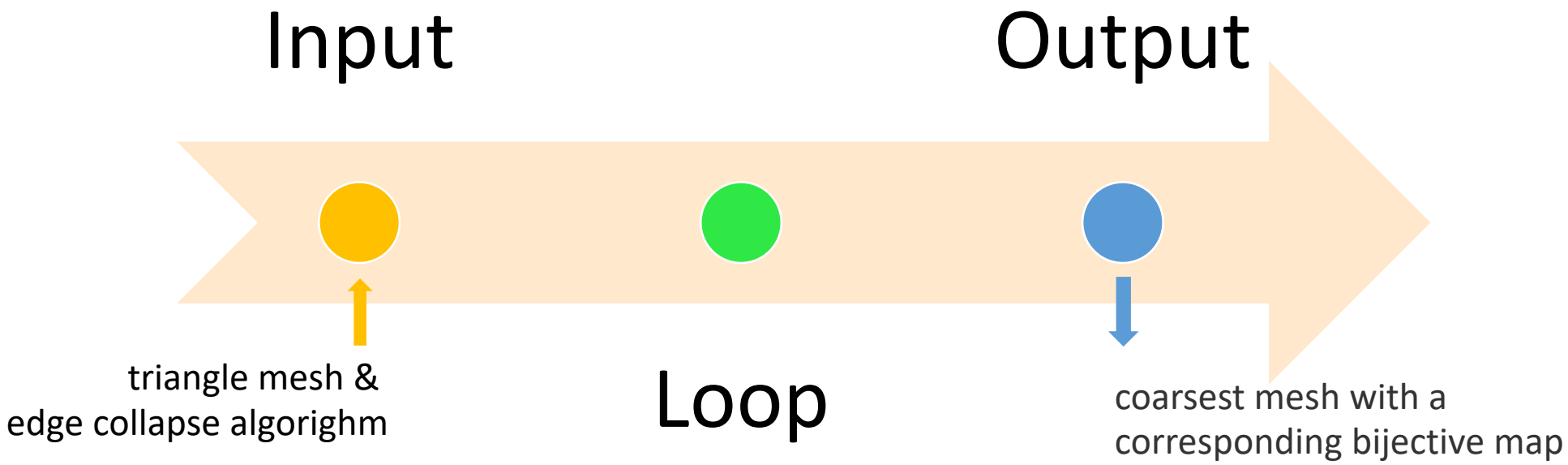
point-to-mesh distance



Data Generation

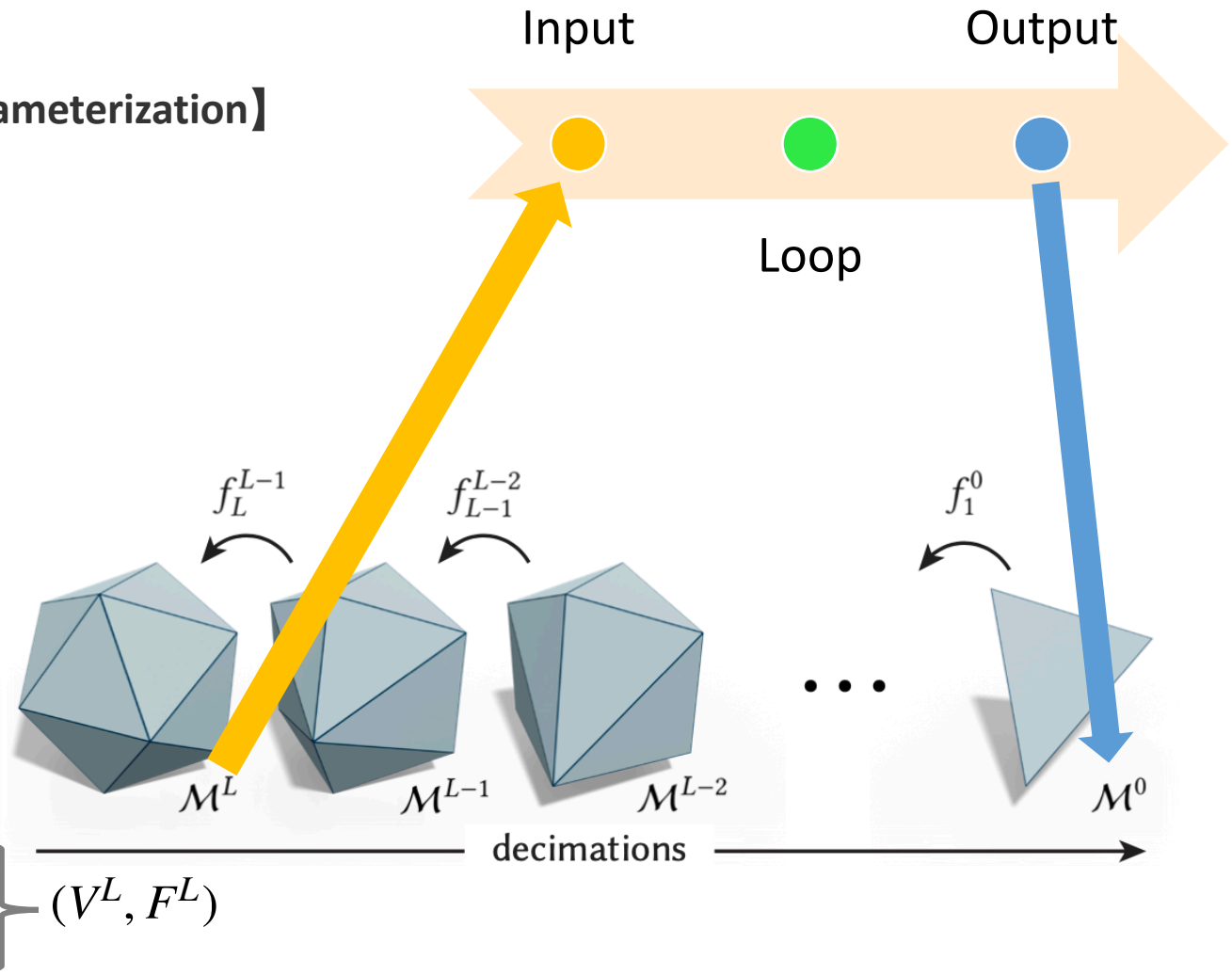
【Successive Self-Parameterization】

- construct a map between different discretizations of the same shape
- combine MAPS and successive mapping



Data Generation

【Successive Self-Parameterization】



$N(j,k) = N(j) \cup N(k)$

Data Generation

【Single Edge Collapse】

- construct a map between different discretizations of the same shape
- combine MAPS and successive mapping

$N(i)$ neighboring vertices of a vertex i

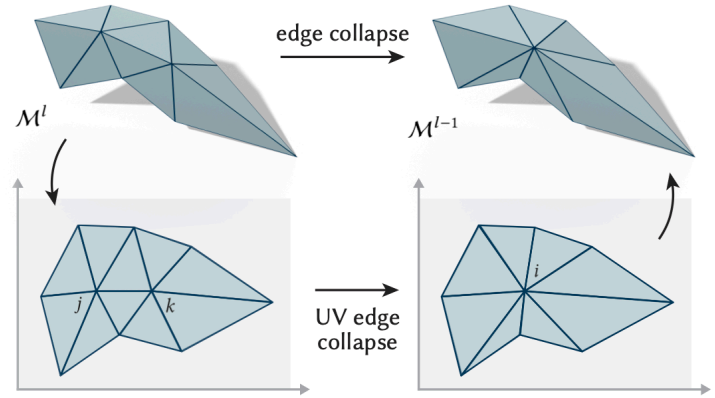
$N(j,k) = N(j) \cup N(k)$ neighboring vertices of an edge (j,k)

Data Generation

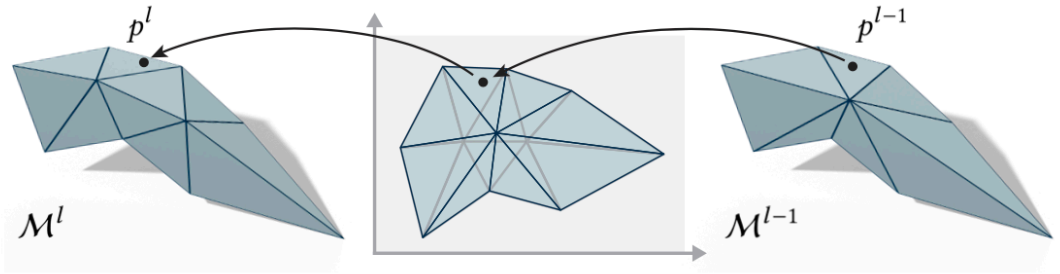
【Single Edge Collapse】

$N(i)$ neighboring vertices of a vertex i

$N(j,k) = N(j) \cup N(k)$ neighboring vertices of an edge (j,k)

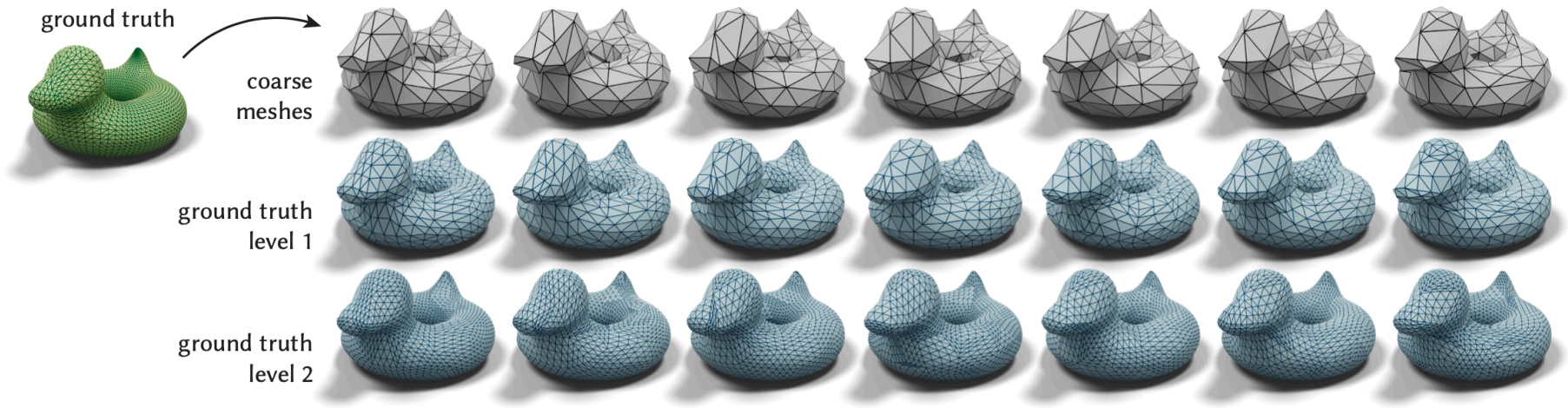


1. parameterize the neighborhood $N(j,k)$ into 2D
2. perform the edge collapse both on the 3D mesh and in UV space
3. boundary vertices of $N(j,k)$ before the collapse become the boundary vertices of $N(i)$ after the collapse



- ✓ Edge collapse algo. -> O(NlogN)
- ✓ Flatten -> constant cost
- ✓ Total successive self-parameterization -> O(NlogN)

Training and Loss



1. use QSLIM with a random sequence of edge collapses to construct several different decimated models
2. plug in self-parameterization to obtain a high-quality bijective map for each coarse and fine pair
3. use Loop topology to retrieve the correspondences
4. use barycentric coordinates b on the coarse mesh to obtain $f(b)$ on the fine mesh
5. use l^2 distance $||f(b) - \in (b)||_2$ to measure per-vertex loss

Network Architecture

operate over atomic **local** mesh neighborhoods and predict differential features

→ invariant

Model I is applied to the 1-ring neighborhood of every vertex to map to a *high-dimensional feature vector*

Use V to predict vertex features based on 1-ring neighborhood

Use the module $\in$

Network Architecture

Initialization Step

Vertex Step

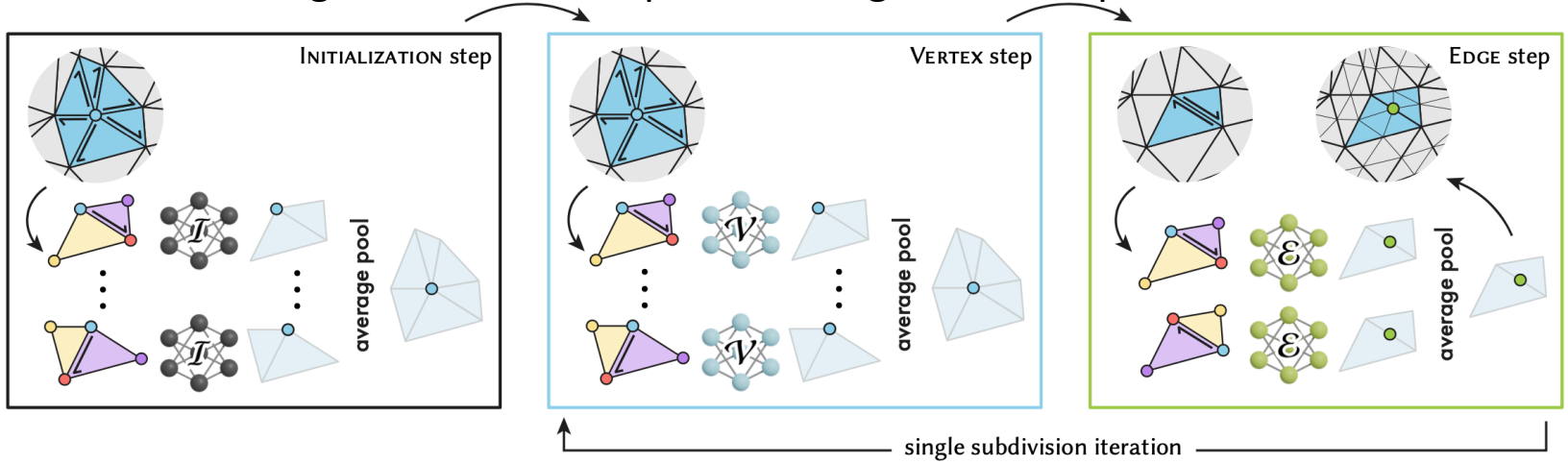
Edge Step

input: mesh at a previous level of subdivision + topological update rule

I: map input per-vertex features to high-dimensional feature vector at each vertex

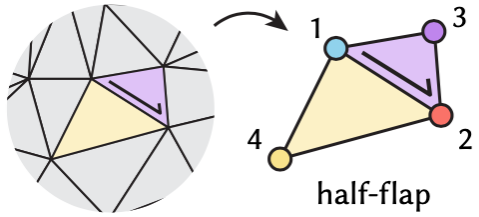
V: in each subdivision iteration, update features at corners of triangles of the input mesh

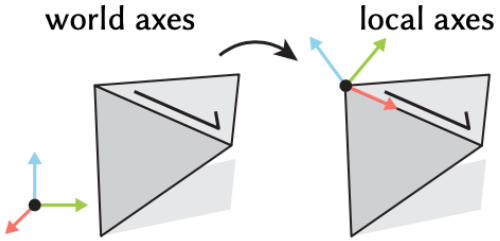
$\in$: compute features of vertices generated at mid-points of edges of the input mesh



Network Architecture

【Half-Flap】
Component of neural model

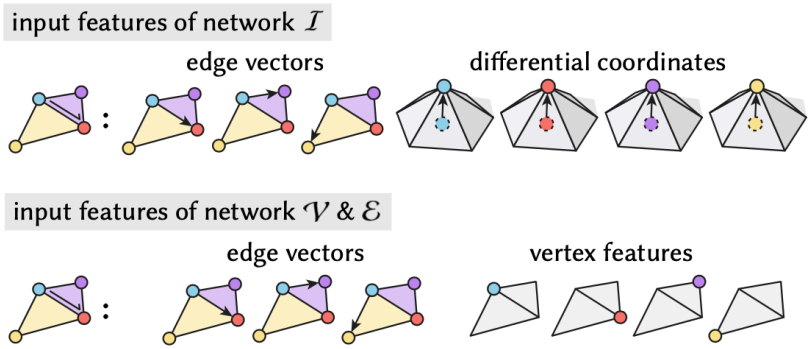




- x-axis: half-edge direction
- z-axis: average two adjacent face normal to get edge normal
- y-axis: cross product

[Input]

- I : three edge vectors and differential coordinates of each vertex(discrete curvature information)
- V and ∈ : edge vectors and per-vertex high-dimensional learned features



[Output]

- V and ∈ : high-dimensional learned features and differential quantities that cna be used to reconstruct the vertex position

Network Architecture

【Classical Subdivision Algorithm】

- even vertices from previous iterations
- newly inserted odd vertices
- different: paper apply V and $\in$ in sequence, instead of parallel. in order to harness neighborhood information from previous steps

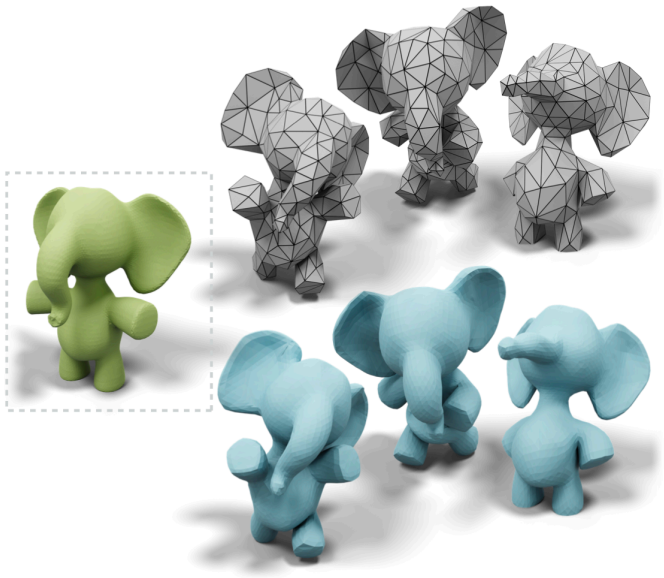
【Design choices】

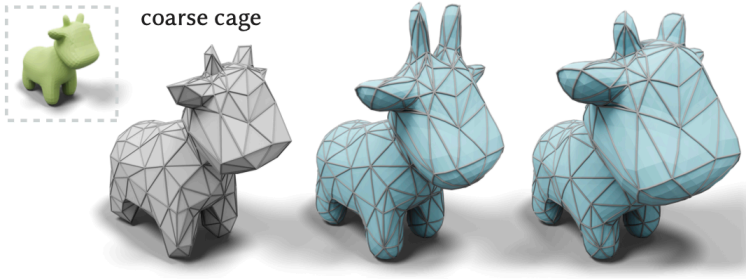
- operate over local mesh patches and share weights rather than global => even a single training pair provides many local mesh patches to train our neural modules
- mesh discretization: don't require re-parameterizing or re-sampling the surface
- represent vertices using differential quantities to a local coordinate frame instead of using global coordinates => invariant

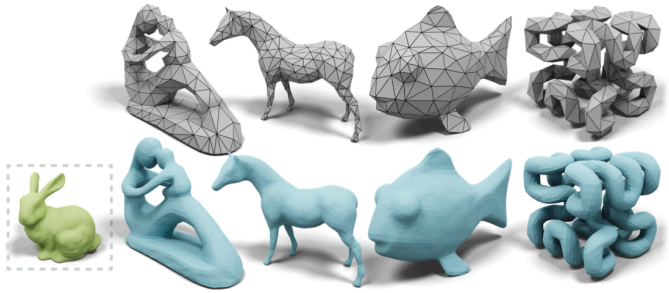
Table of Content

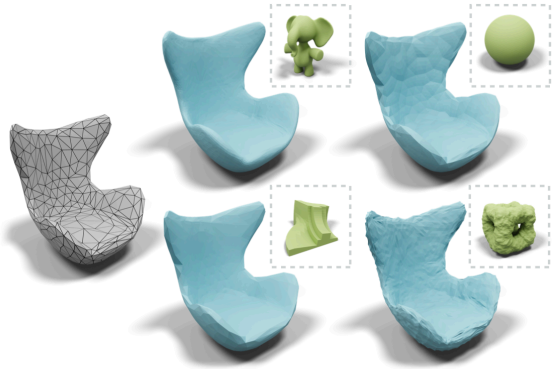
Evaluations

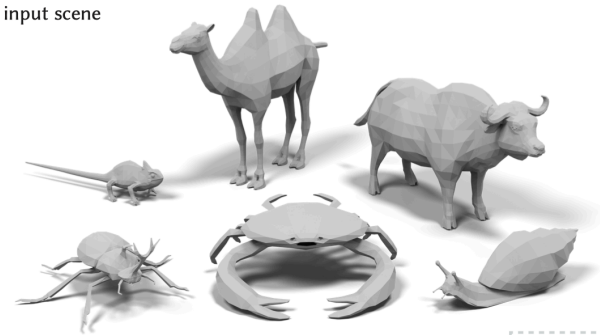
- isometric deformations
- non-isometric deformations
- shapes from different classes
- shapes from different types of distretizations
- multiple shapes and categories

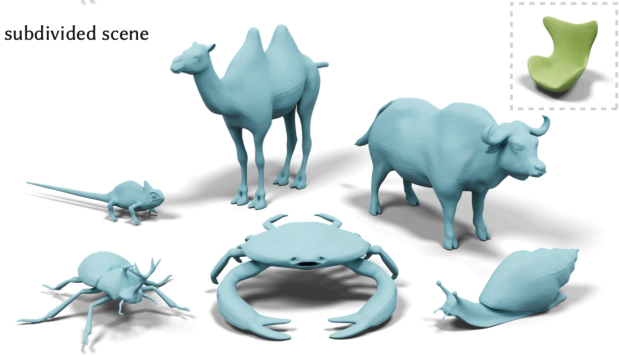










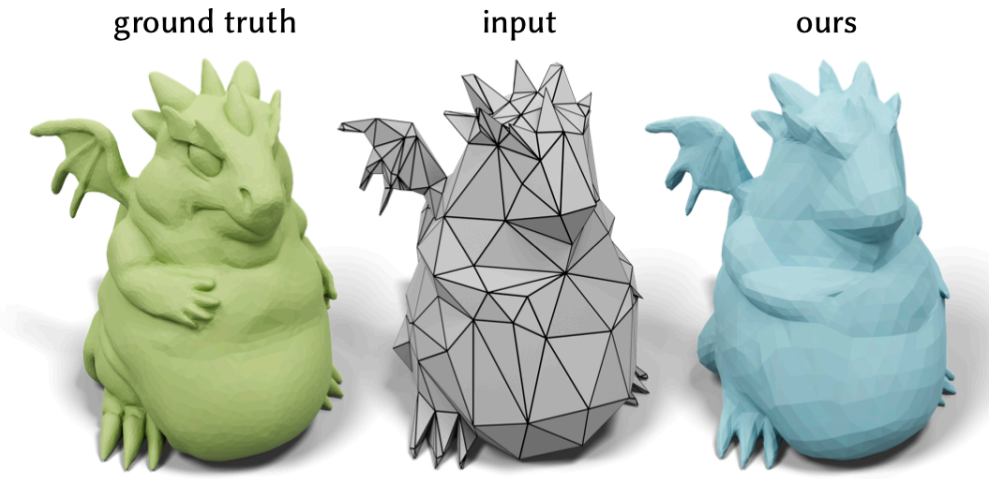


Evaluations

- non-linear subdivision beyond simple linear averaging used in classical techniques
- only requires a set of high-resolution meshes for learning network weights
- output is a surface mesh with deterministic connectivity based on the input, enabling direct use in the standard graphics pipelines such as texture mapping
- ensure rotation and translation invariance(encode vertex position data in a local frame rather than the entire shape)
- refine the mesh locally, input could be arbitrary
- don't require co-aligned training data with a well-defined object space
- output is translation and rotation invariant(describe in local coordinate system)

Limitations & Further Work

- Non usage of global information
- Non usage information from a wider neighborhood and to dive to a deeper subdivision level



- quadrilateral meshes
- surface with boundaries