Neural Subdivision

> https://www.dgp.toronto.edu/~hsuehtil/pdf/neuralSubdiv_95mb.pdf

Intro

- **what is neural subdivision?** Neural Subdivision, a novel framework for data-driven coarse-to-fine geometry modeling
  - Neural subdivision避免了经典subdivision的缺陷：
    - volume shrinkage
    - over-smoothing
    - amplification of tessellation artifacts
- **该论文的创新性**：
  - 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 vettex position data in a local frame rather than the entire shape)

Background

(经典)

- subdivision surface: recursive up-sampling of a discrete surface mesh

- classic methods:
  - each input mesh element is divided into many vertices
    - split edges
    - add vertices
  - smooth positions of the mesh vertices by taking a **linear weighted average** of their neighbors's positions(weight scheme based purely on the local mesh connectivity)
- 应用：
  - 交互式建模 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
- 缺陷：
  - 容易overly smooth the entire shape
  - do not identify details to maintain细节丢失
  - 全局使用 fixed on-size-fits-all weighting rules 丢失了大量信息的使用

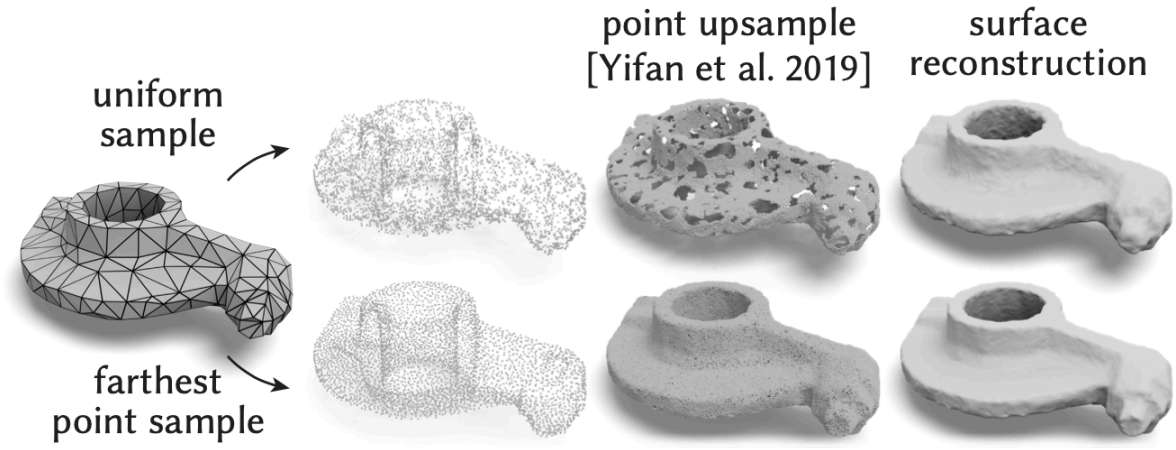
(神经)

- **basic process:**
  - input: coarse triangle mesh
  - Loop subdivision: recursively subdivide by applying the fixed topological, but predicting vertex positions using a neural network conditioned on the local geometry of a patch
- 问题：如何collect training data pairs? 虽然现在3D模型很多，但是都不包含需要的信息
  - 解决：在训练时，仅对 origin / 连接性高的曲面网格进行自我监督；随机生成训练样本的低分辨率版本，同时保持其表面间的双射

2.2 Neural Geometry Learning

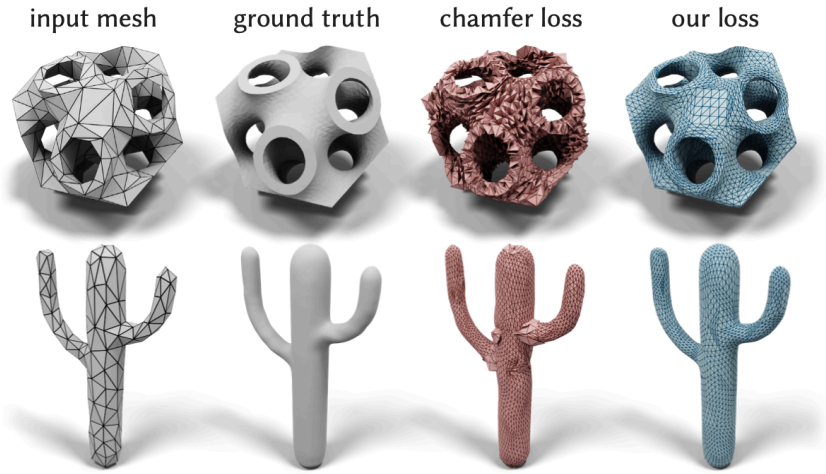
（寻找神经网络的输入，还是上面这个问题）

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



- deform global template
  - predict vertex coordinates
  - warps the entire 3D domain conditioned on a latent vector that encode the deformation target
  - 缺点
    - output is limited to deformations of a single shape
- others
  - using a local or global parameterization to unfold a mesh into 2D grid
  - apply graph-based techniques adapted for meshe
- 论文的优点
  - 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)
- 论文采用MeshCNN的变形
    - **MeshCNN** learn filter over the local mesh structure via undirected edges -> deterministic tasks
    - 论文正好相反 generative tasks, develop features over the half-flaps(an edge along with its two adjacent triangles)[each half-flap has a canonical orientation, 对运动不变性很有帮助]
- loss距离
    - surface-to-surface distances(chamfer distance): define via closest-point queries
        - 问题：match many points to the same point, while leavint other points unmatched, resultiing in self-overlaps and unrepresented areas



3.Neural Subdivision Overview

- **training and loss**
    - train with l^2 loss: distance between each predicted vertex position at every level of subdivision and its corresponding point on the original shape
    - dataset: (no existing dataset)develop a training data comprising of coarse and fine meshes tieh bijective mappings between

them

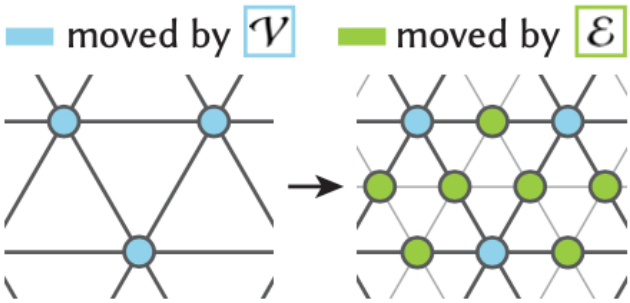
- **data-generation**
  - each subdivided mesh at any level can be mapped back to the initial coarse mesh
    - mid-point-to-vertex
    - vertex-to-vertex
  - 坐标转换
    - v: subdivision新生成的点
    - g: subdivided-to-coarse bijective mapping
    - f: coarse-to-origin bijective mapping

      > 通过随机选边和点的方式将origin进行coarse，得到一序列coarse model
      >
      > computing a conformal map between the 1-ring edge neighborhood (before the collapse) and the 1-ring vertex neighborhood (after the collapse)

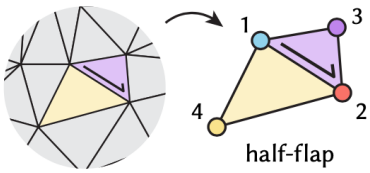
    - $f(g(v))$: unique point on the original mesh corresponding to v
- **network architecture**
  - operate over atomic local mesh neighborhoods and predict differential features(represent geometry in the local coordinates not in world coordinates)
  - **Initialization step**
    - compute differential pre-vertex quantities based on local coordinate
    - model I is applied to the 1-ring neighborhood of every vertex to map to a *high-dimensional feature vector*

> encode local geometry of the patch, and differential quantities which directly represent local geometry to reconstruct the vertex coordinates

- split edges at midpoint, subdivide each triangle into 4



- **Vertex step**
  - use V to predict vertex features based on 1-ring neighborhood
- **Edge step**
  - use the module ε
- half-flap
  - directed edge: define the local coordinate frame
  - concatenate 4 vertices' features and feed them into shallow MLP(multi-layer perceptrons)



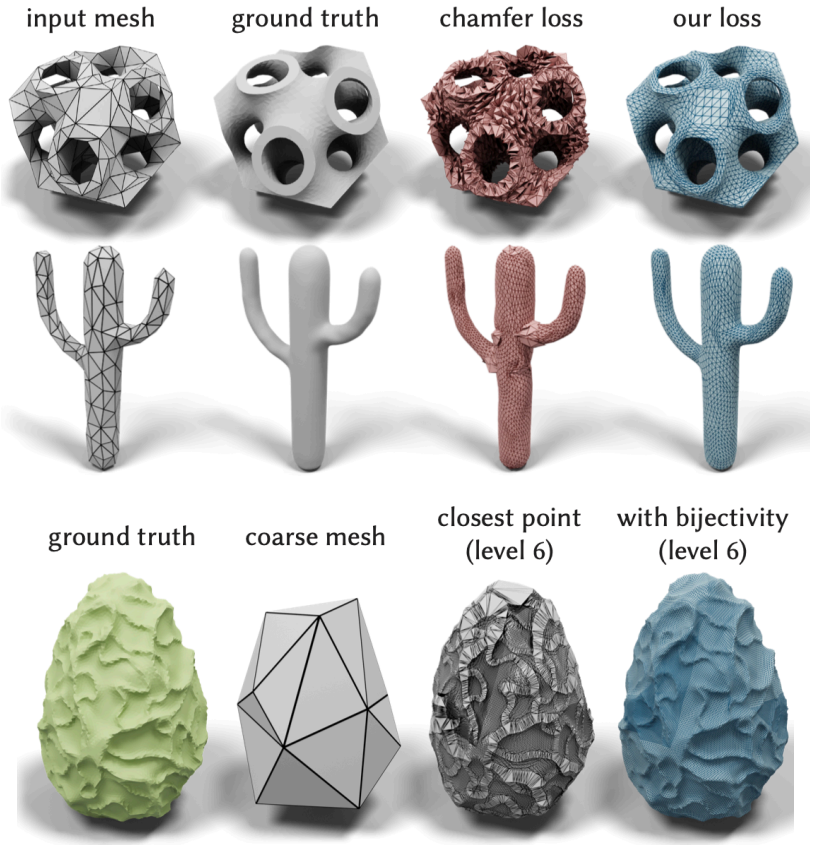
  - I, V process all half-flaps, use average pooliing to combine the half-flap features into per-vertex features; ε combine feature from two half-flaps(边的两个方向) via average pooling

---

4.Data Generation and Training

- **naive subdivision training**

    - 随机取样pairs of coarse/fine meshes
    - measure the disatance between the network's predicted subdivision and the ground truth
    - iterate over coarse/fine pairs while optimizing the loss
    - **缺陷**
        - chamfer-like loss or point-to-mesh distance will lead to incorrect and self-overlapping matehes

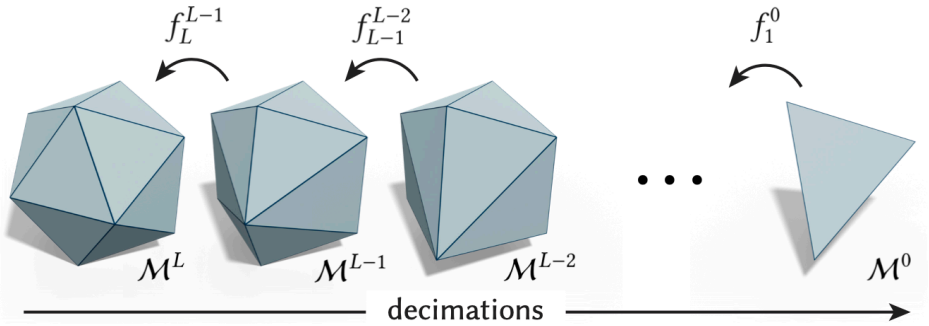


- **Successive Self-Parameterization**
    - construct a map between different discretizations of the same shape
    - combine MAPS and successive mapping
    - **two-step module**
        - input: a triangle mesh and an edge collapse algorithm
        - output: decimated mesh with a corresponding bijective map between the input and the decimated model

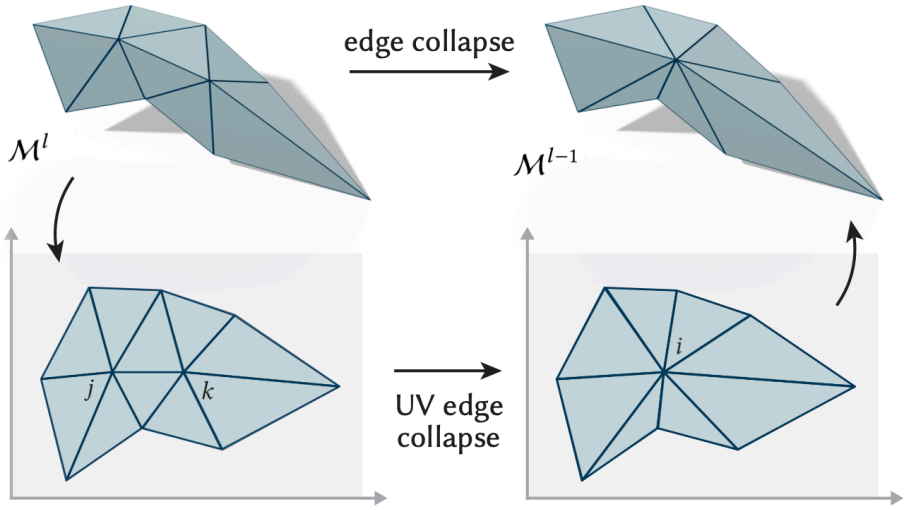
We denote the input triangle mesh as $\mathcal{M}^L = (\mathsf{V}^L, \mathsf{F}^L)$, where $\mathsf{V}^L, \mathsf{F}^L$ are vertex positions and face information respectively at the original level L. The input mesh $\mathcal{M}^L$ is successively simplified into a series of meshes $\mathcal{M}^l = (\mathsf{V}^l, \mathsf{F}^l)$ with $0 \leq l \leq L$, where $\mathcal{M}^0 = (\mathsf{V}^0, \mathsf{F}^0)$ is the coarsest mesh. For each edge collapse $\mathcal{M}^l \rightarrow \mathcal{M}^{l-1}$, we compute the bijective map $f_l^{l-1} : \mathcal{M}^{l-1} \rightarrow \mathcal{M}^l$ (see Fig. 12) on the fly. The final map $f_L^0 : \mathcal{M}^0 \rightarrow \mathcal{M}^L$ is computed via composition,

$$f_L^0 = f_L^{L-1} \circ \cdots \circ f_1^0. \tag{1}$$

We now focus our discussion on the computation of a bijective map for a single edge collapse.



- **Single Edge Collapse**
  - terms
    - $N(i)$: neighboring vertices of a vertex i
    - $N(j,k) = N(j) \cup N(k)$: neighboring vertices of an edge (j,k)
  - compute the bijective map for the edge's 1-ring $N(j,k)$
    1. parameterize the neighborhood $N(j,k)$ into 2D
    2. perform the edge collapse both on the 3D mesh and in UV space



- boundary vertices of $N(j,k)$ before the collapse become the boundary vertices of $N(i)$ after the collapse

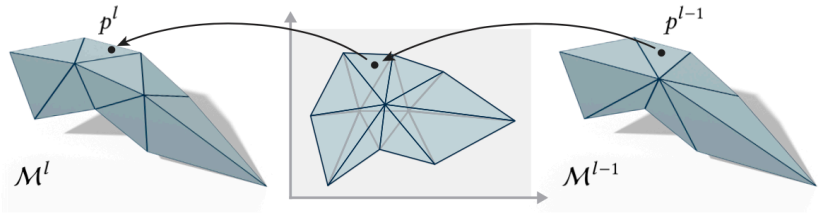
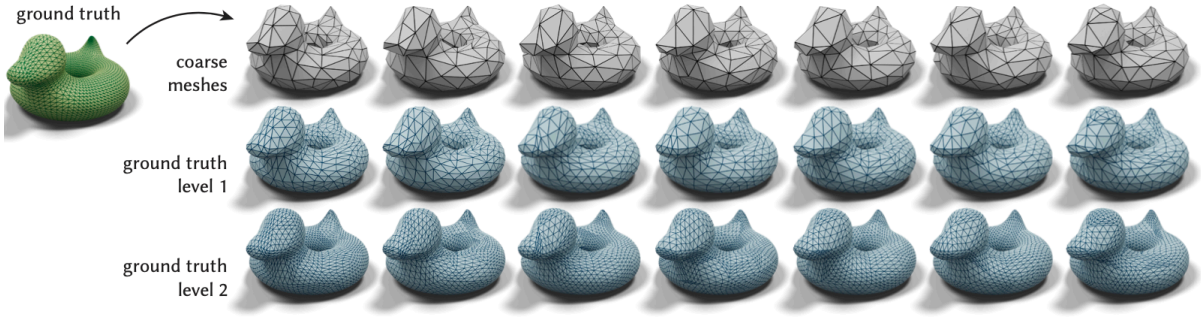


Fig. 14. Since both the pre-collapse and post-collapse parameterizations of the 1-ring map it into the same 2D domain, we can easily use the shared UV space to map a point back and forth between $\mathcal{M}^l$ and $\mathcal{M}^{l-1}$.

  - use conformal flattening to compute the UV parameterization

- **4.3 Implementation**
  - edge collapse algo. -> $O(NlogN)$
  - flatten -> constant cost
  - total successive self-parameterization -> $O(NlogN)$
- **4.4 Training Data & Loss Computation**



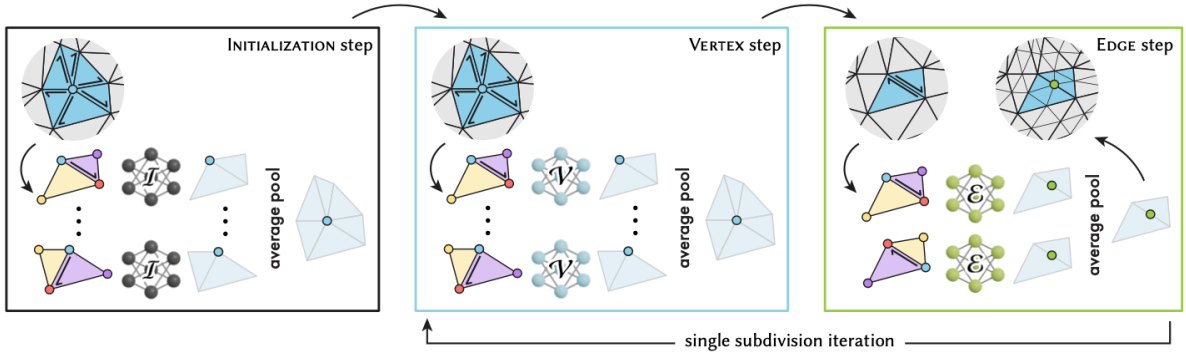
1. use QSLIM with a random sequence of edge collapses to construct several different decimated models

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 $||f(b) - \varepsilon(b)||_2$ to measure per-vertex loss

---

5.Network Architecture

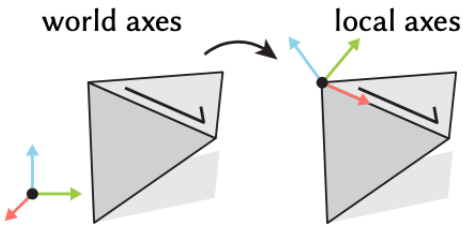
- **INITIALIZATION step**
  - input: mesh at a previous level of subdivision + topological update rule(Loop use mid-point subdivision)
  - I : map input per-vertex features to high-dimensional feature vector at each vertex
- **VERTEX step**
  - V: in each subdivision iteration, update features at corners of triangles of the input mesh
- **EDGE step**
  - ε: compute features of vertices generated at mid-points of edges of the input mesh



- classical subdivision algorithm(在此基础上改进)
  - even vertices from previous iterations
  - newly inserted odd vertices
  - different: paper apply V and ε in sequence, instead of parallel.

in order to harness neighborhood information from previous steps

- Design choices(及时训练集很少也能使效果不错)
  - 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
- component of neural model: half-flap 优点
  - provide a unique canonical orientation for the 4 vertices
  - provide well-defined local coordinate fram
  - each is a shallow MLP -> define over features of four ordered points
  - train operators per model (I, V, ε) across all levels of subdivision and training examples
- **average pooling**: aggregate features from different half-flaps to per-vertex features
  - 1,2 step apply the half-flap operator to every outgoing edge in a 1-ring neighborhood of a vertex, and average pooling aggregates per-half-flap outputs into a per-vertex feature
  - 3th step apply half-flap operator for each directions of the edge and again uses average pooling to get the vertex feature
- **representation**
  - local differential quantities in local frame of each half-flap => ensure invariance to righd transformation
    - x-axis: half-edge direction
    - z-axis: average two adjacent face normal 得到的边缘法线
    - y-axis: cross product



- input
  - I: three edge vectors and differential coordinates of each vertex(discrete curvature information)
  - V and ε: edge vectors and per-vertex high-dimensional learned features
- output
  - V and ε: high-dimensional learned features and differential quantities that cna be used to reconstruct the vertex position

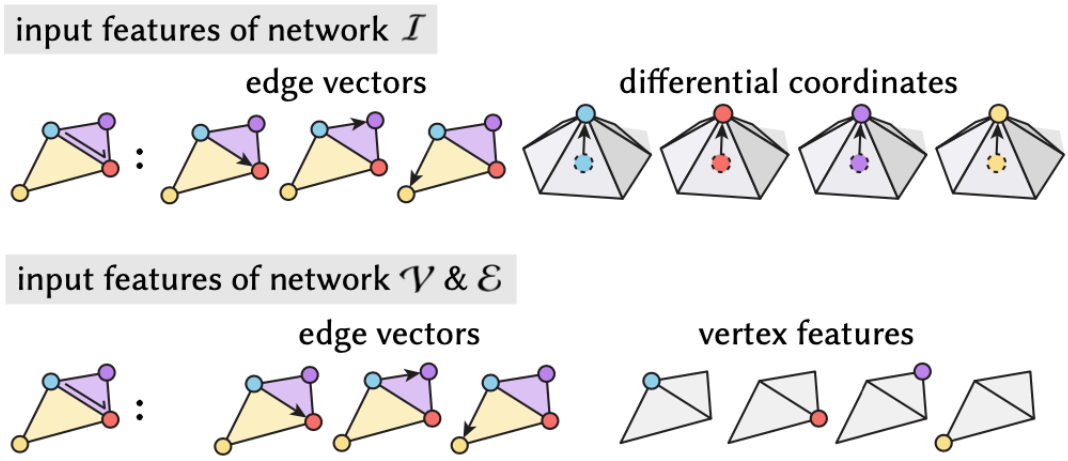


Fig. 19. The input feature to module $\mathcal{I}$ consists of three edge vectors from the source vertex (blue) and vectors of the differential coordinates for the four vertices. The input features to module $\mathcal{V}$ and $\mathcal{E}$ are three edge vectors with per-vertex high-dimensional features from the previous steps.

6.Evaluations

- **isometric deformations**: train on a single pose => generalize to unseen poses under isometric deformations
- **non-isometric deformations**

- **shapes from different classes**
- **shapes from different types of distretizations**
- **multiple shapes and categories**

7.Limitations & Future Work

- extension
  - quadrilateral meshes
  - surface with boundaries
- limitation
  - 整合global information
  - harness information from a wider neighborhood and to dive to a deeper subdivision level

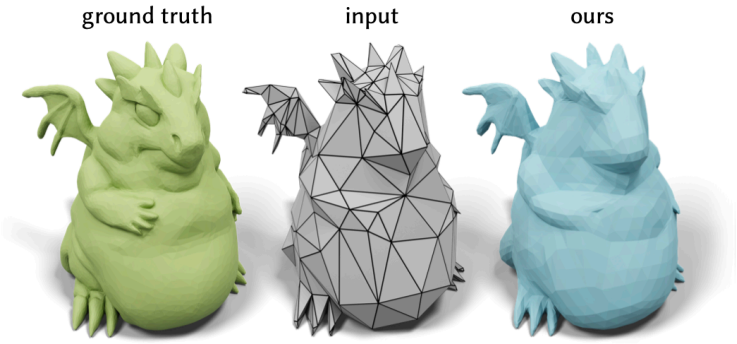


Fig. 30. Our approach is based on local geometry, and thus fails to hallucinate semantic features. ©Bratty Dragon by Splotchy Ink under CC BY.