GEOMetrics

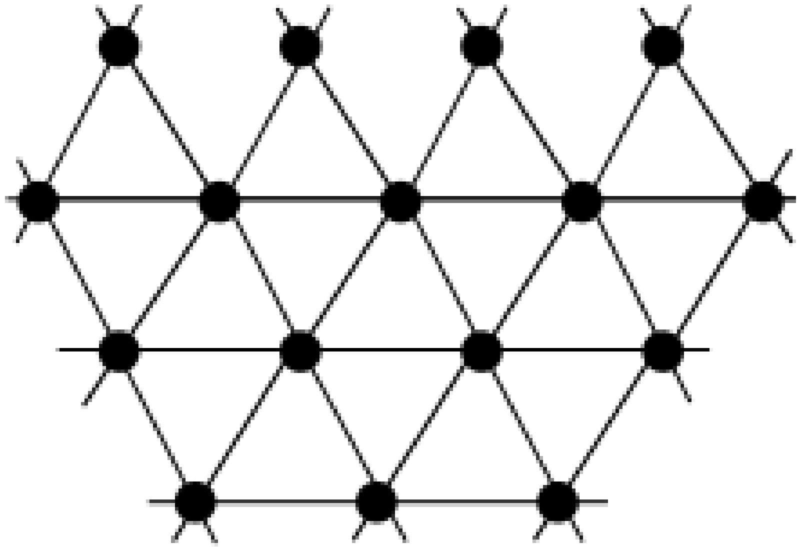
Exploiting Geometric Structure for Graph-Encoded Objects

Edward Smith, Scott Fujimoto, Adriana Romero, David Meger

facebook AI Research

Topic: Mesh Object Generation

What is a Mesh?



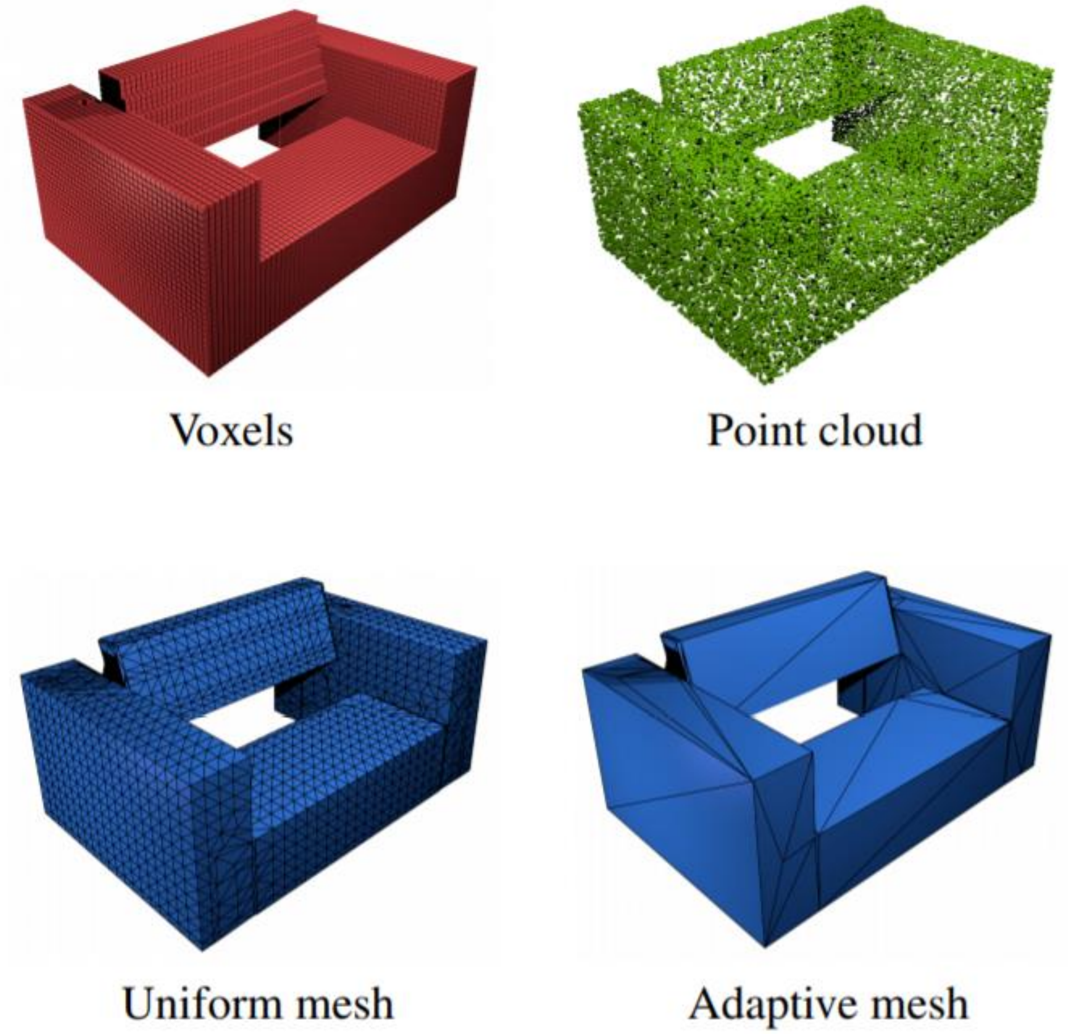
3D surface representation

- Collection of connected triangular faces

Defined by a graph G = {V, A}

- V = collection of verticies
- A = Adjacency Matrix
  - A[i,j] != 0 if and only if there exist a face f, such that {i,j} is in f

Why Choose Meshes?



Mesh Generation

How do you predict a complicated graph structure?

- You don't

Deform a predefined mesh

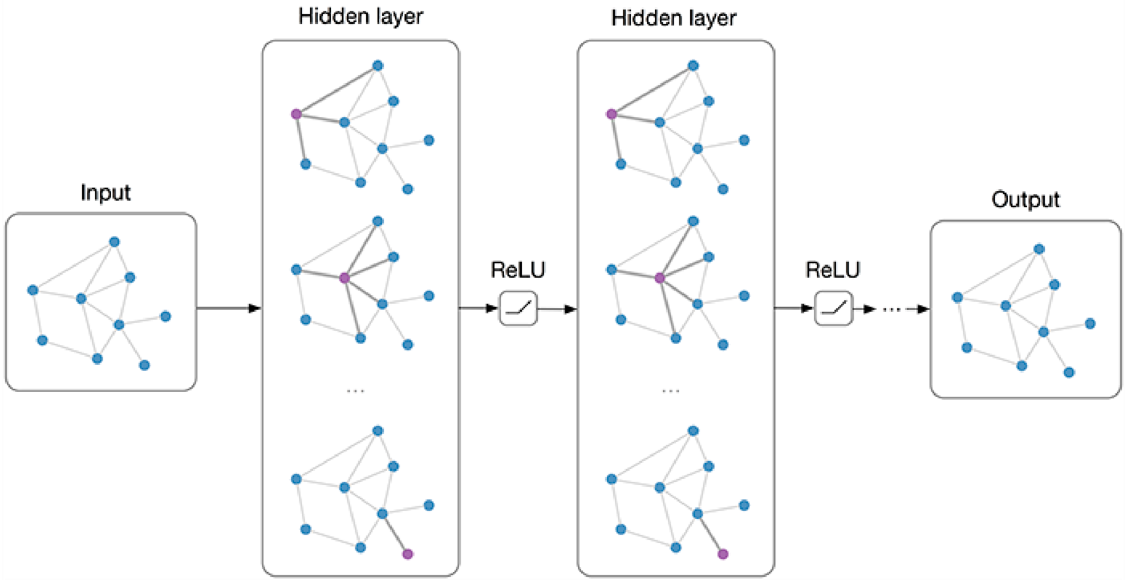
- Assume initial graph structure
- Predict updates to the structure
- How do we make these updates?
- How do we compare to know mesh ground truth?

Deform: Graph Convolutional Network

Input: - graph {V, A} - features over V, {H} - weight and bias {W,b}

Apply the following operation:

$$H' = \sigma (AHW + b)$$



Problem:

- Vertex smoothing
- Each vertex in an mesh is important
- Exacerbated in adaptive mesh

Reference for figure https://tkipf.github.io/graph-convolutional-networks/:

Solution: Zero Neighbor GCN

Basic formulation: $H' = \sigma \left(AHW + b \right)$

Higher order : $H' = \sigma \left(\left[AH_1 \| A^2 H_2 \| \ldots \| A^k H_k \right] W + b \right)$

0N-GCN: $H' = \sigma \left(\left[A^0 H_0 \| AH_1 \| \right] W + b \right)$

$H' = \sigma \left(\left[H_0 \| AH_1 \| \right] W + b \right)$

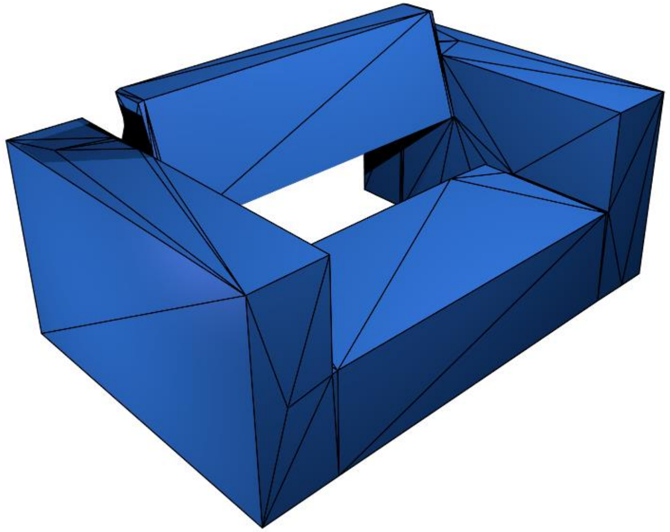
- Soft middle ground between neighbor update and none
- Adaptive meshes should emerge more easily

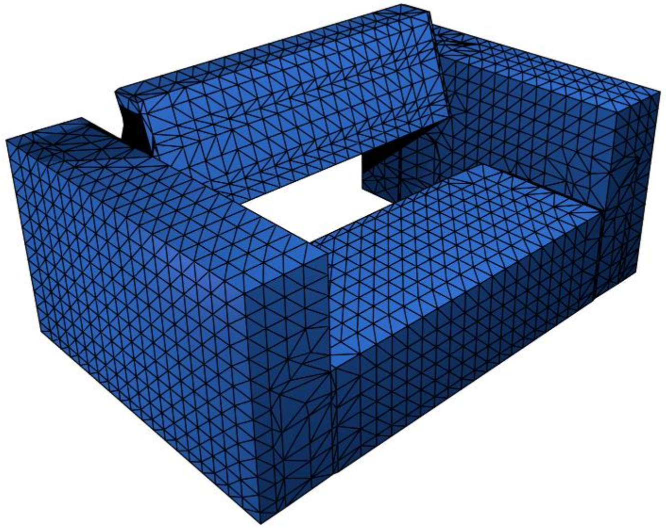
Compare: Chamfer Distance

$$\mathcal{L}_{\text{Chamfer}} = \sum_{p \in S} \min_{q \in \hat{S}} \|p - q\|_2^2 + \sum_{q \in \hat{S}} \min_{p \in S} \|p - q\|_2^2$$

Problem with naïve mesh application:

- Arbitrary vertex placement
- No consideration of the faces they define



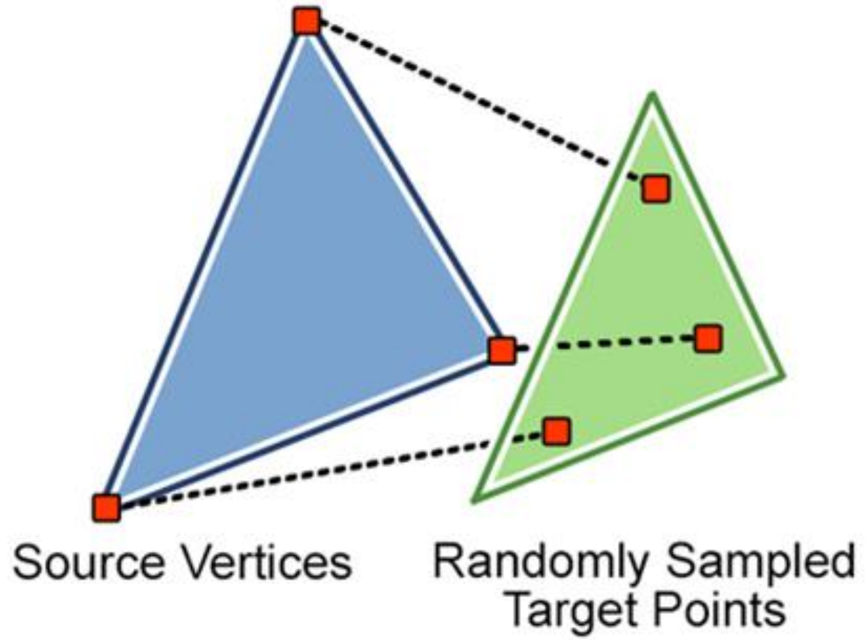


Vertex-to-Point

$$\mathcal{L}_{\text{Chamfer}} = \sum_{p \in S} \min_{q \in \hat{S}} \|p - q\|_2^2 + \sum_{q \in \hat{S}} \min_{p \in S} \|p - q\|_2^2$$

Past attempt to solve issue:

- Make ground truth and predicted meshes huge
- Definitely not going to get adaptive mesh



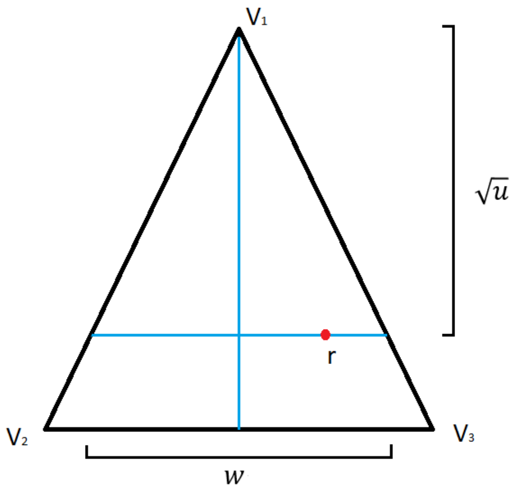
Solution: Sample using Reparameterization

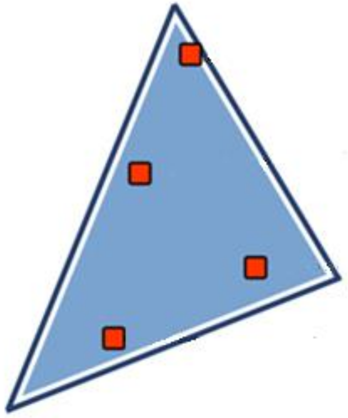
Sample both meshes uniformly

- Given: $F = \{v_1, v_2, v_3\}$, U & W are Uniform(0,1)
- Sample: $u \sim U\ ,\ w \sim W$
- Sample projected onto triangle:

$$r = (1-\sqrt{u})v_1 + \sqrt{u}(1-w)v_2 + \sqrt{u}wv_3$$

- Select faces at rate proportional to relative surface area





Point-to-Point Loss

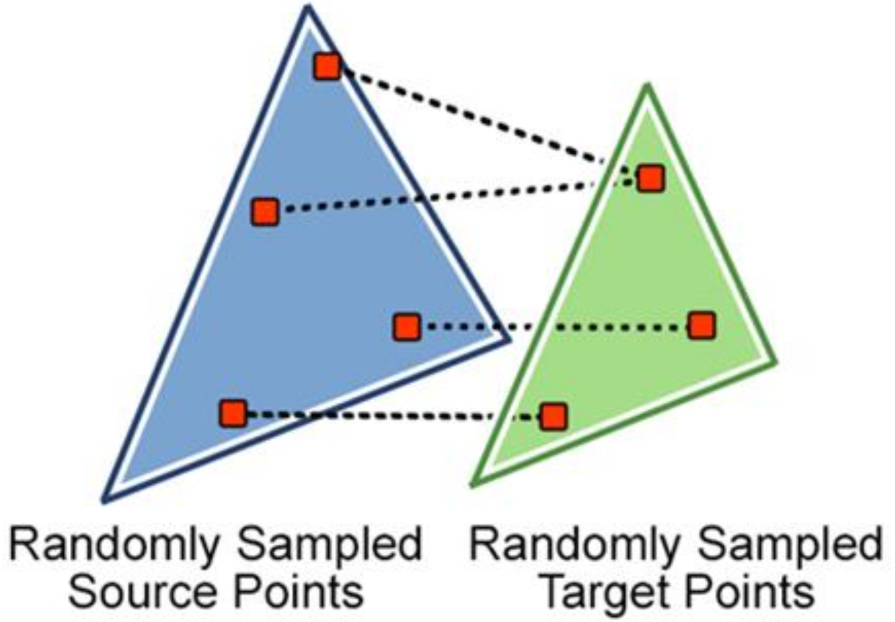
$$\mathcal{L}_{\text{PtP}} = \sum_{p \in S} \min_{q \in \hat{S}} \|p - q\|_2^2 + \sum_{q \in \hat{S}} \min_{p \in S} \|p - q\|_2^2$$

Can sample independent of vertex position

- Removes ambiguity of the target placement
- Do not have to match vertex placement

Face information is now take into account

Vertices can be placed optimally

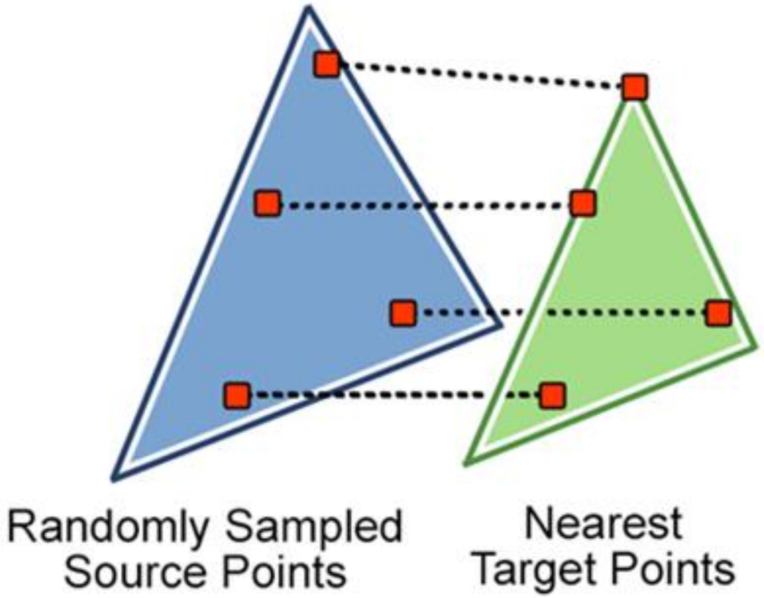


Point-to-surface Loss

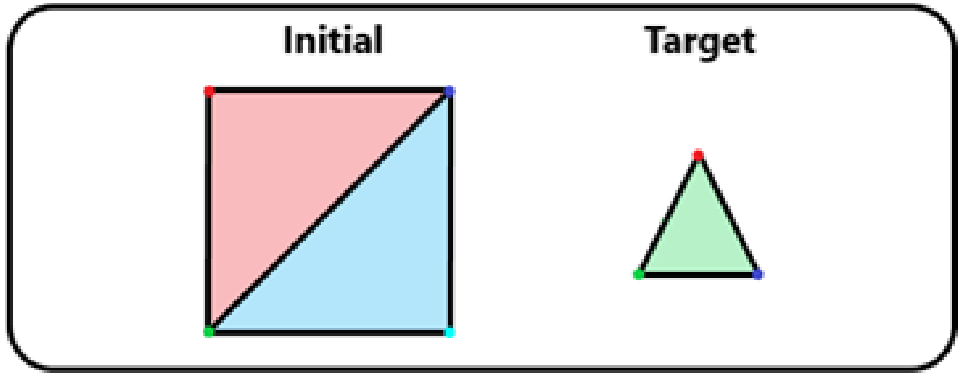
Can do even better still: compare to surfaces instead of points

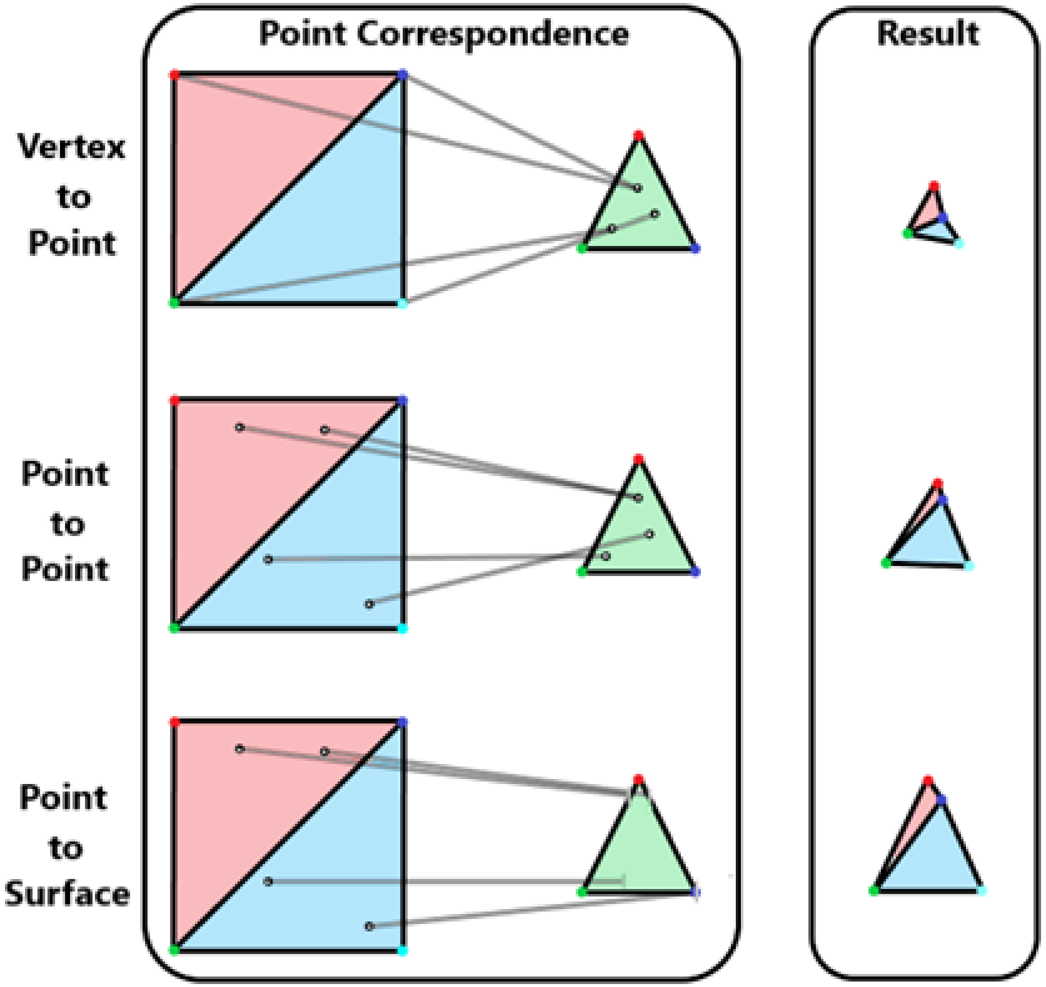
$$L_{\text{PtS}} = \sum_{p \in S} \min_{\hat{f} \in \hat{M}} dist(p, \hat{f}) + \sum_{q \in \hat{S}} \min_{f \in M} dist(q, f)$$

- Function dist() is the minimum distance from a point to a triangle in 3D space
- More accurate to the previous functions

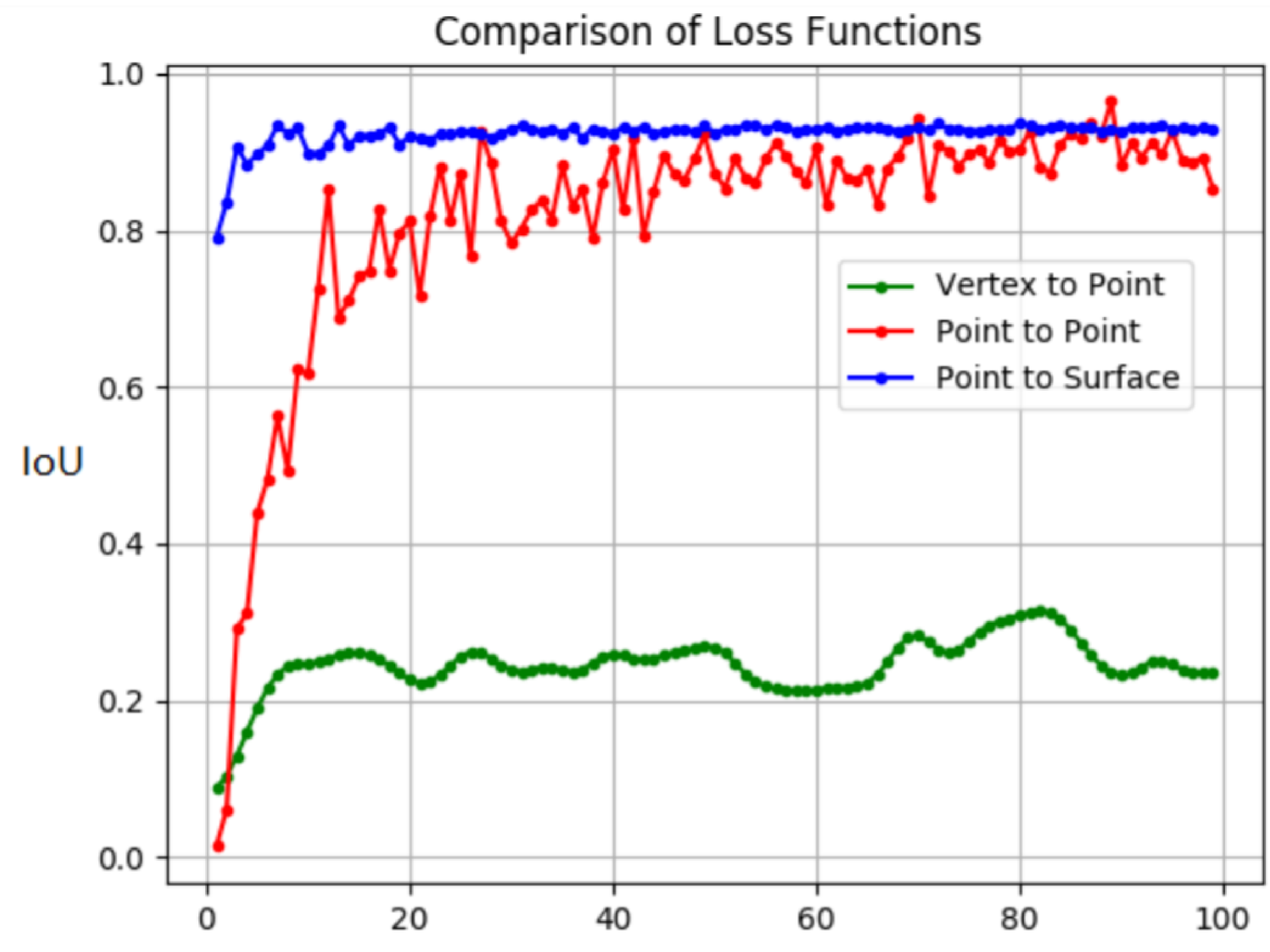


Toy Example


Initial
Target


Point Correspondence
Result
Vertex
to
Point
Point
to
Point
Point
to
Surface

Toy Example

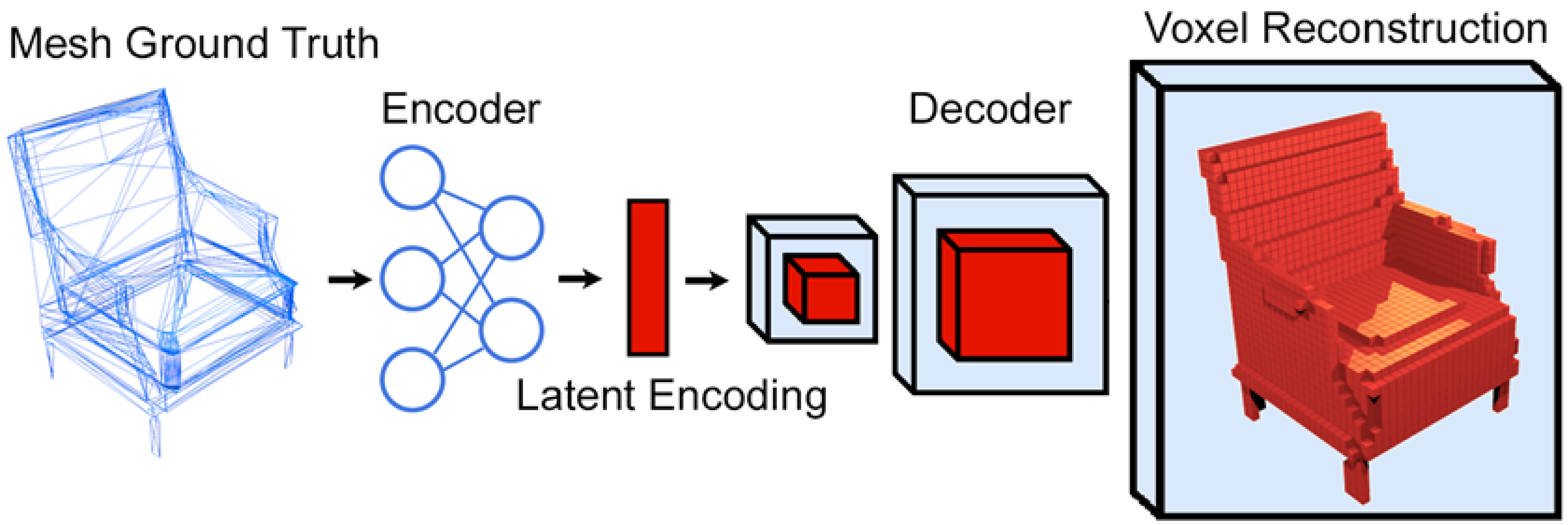


Comparison of Loss Functions
1.0
0.8
0.6
0.4
0.2
0.0
IoU
0
20
40
60
80
100
Vertex to Point
Point to Point
Point to Surface

Latent Loss

Train an encoder decoder system from mesh to voxel space

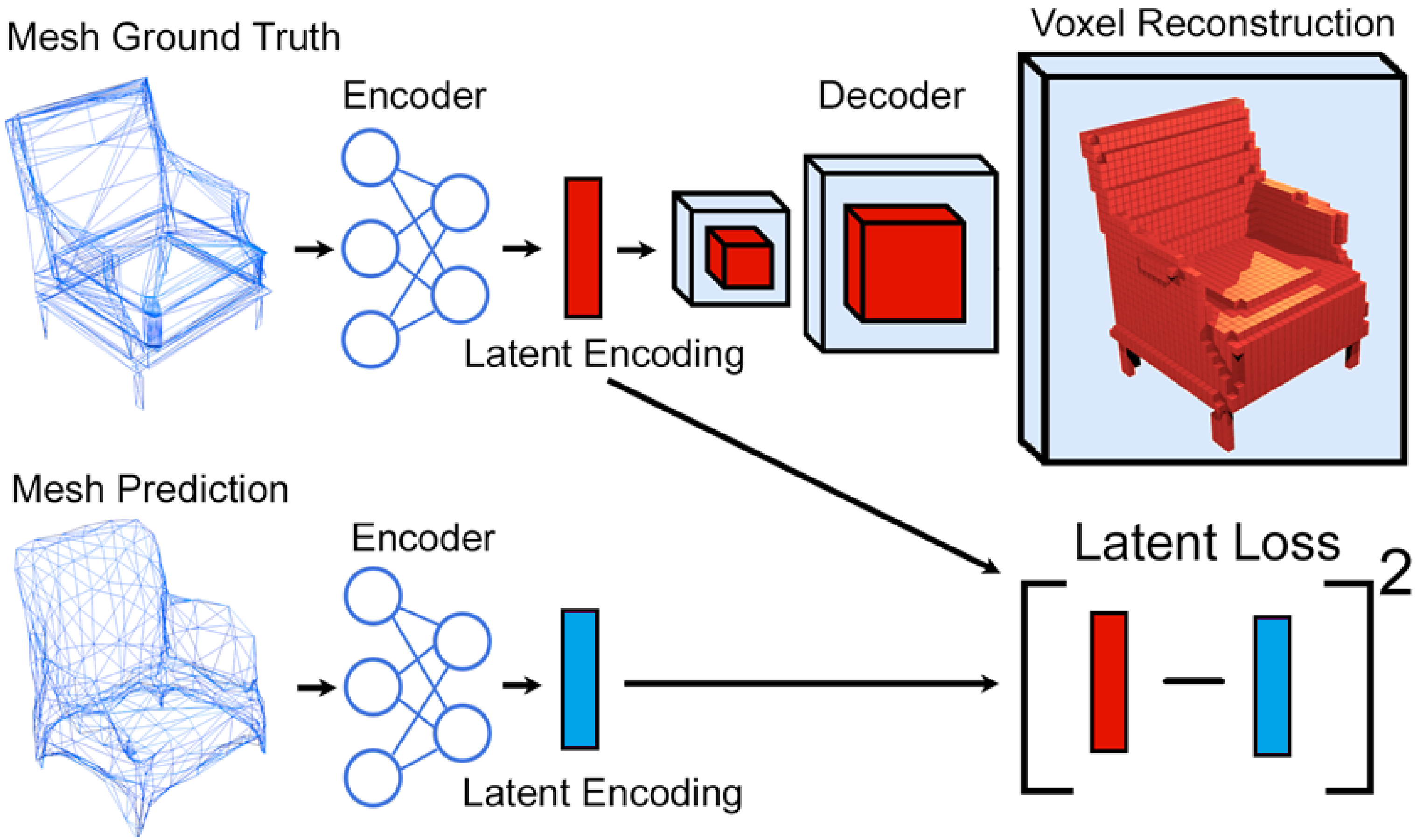
- Using 0N-GCN networks followed by 3D convolutional network



- The latent encoding should poses all info on passed object

Latent Loss

Use the difference between latent encodings of GT and predicted objects as a loss signal:

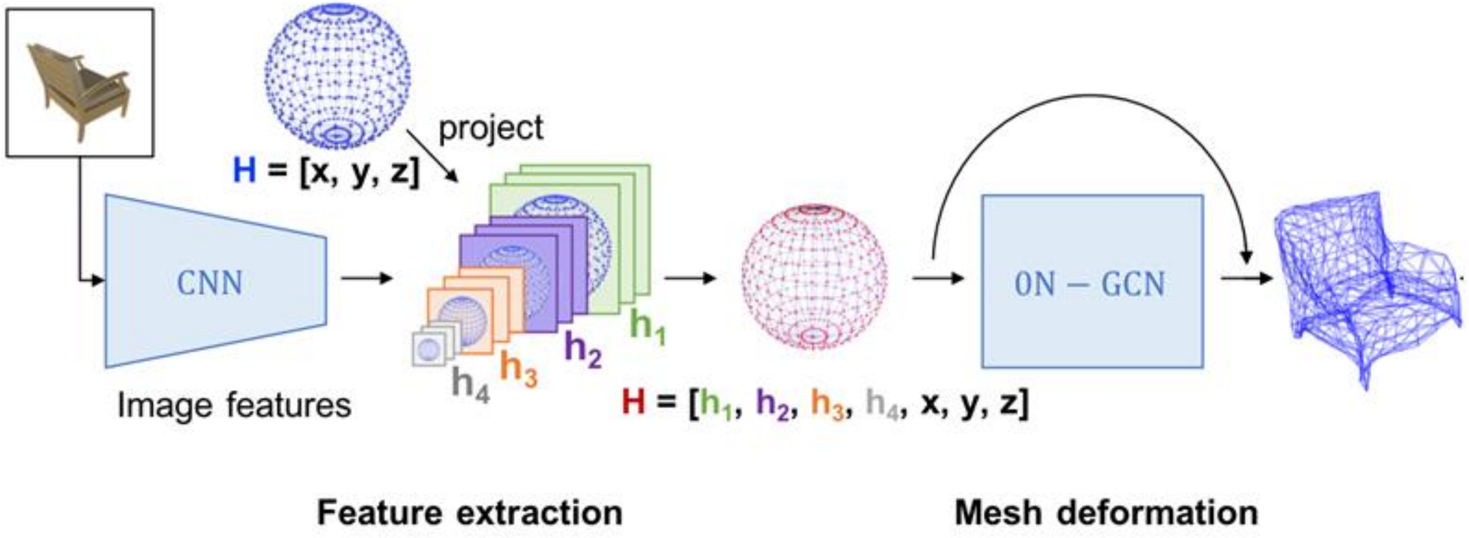


Mesh Generation Pipeline

Input: Image & initial mesh

Output: Mesh reconstruction

1. Pass image through CNN
2. Project image features onto initial mesh as feature vectors
3. Pass through graph through multiple 0N-GCN layers
4. Train using: PtP loss, PtS loss, latent loss

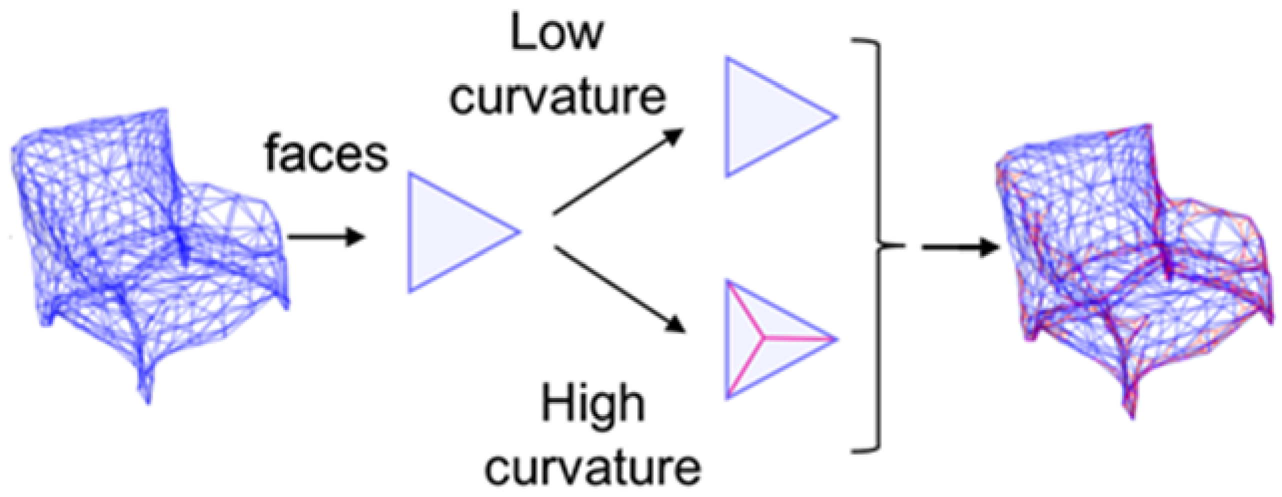


Face Splitting

Analyse local curvature of the mesh

- At each face calculate average change in normal

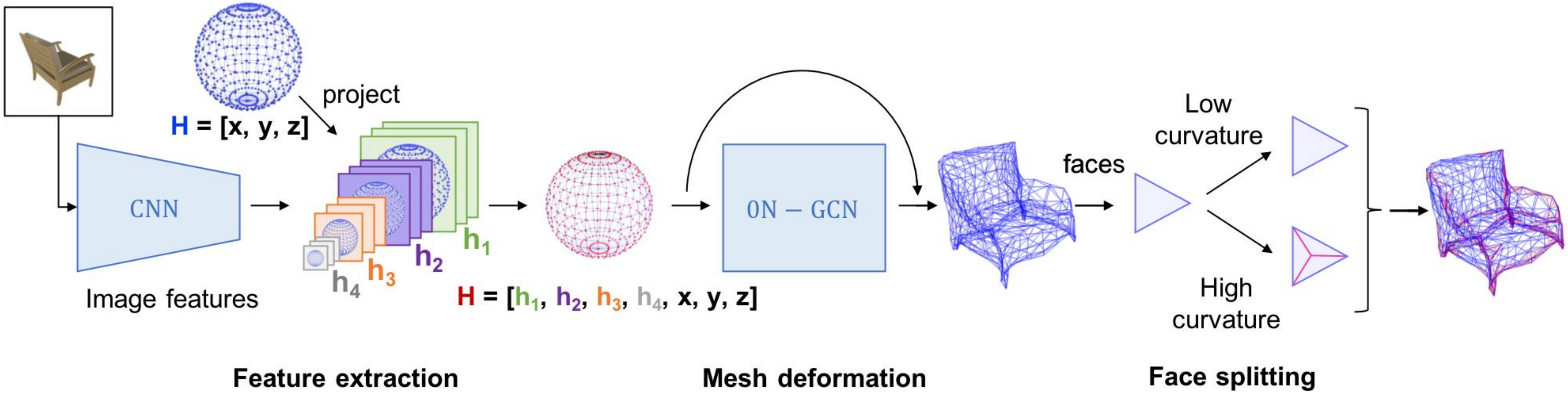
Every face over a given threshold is split into three



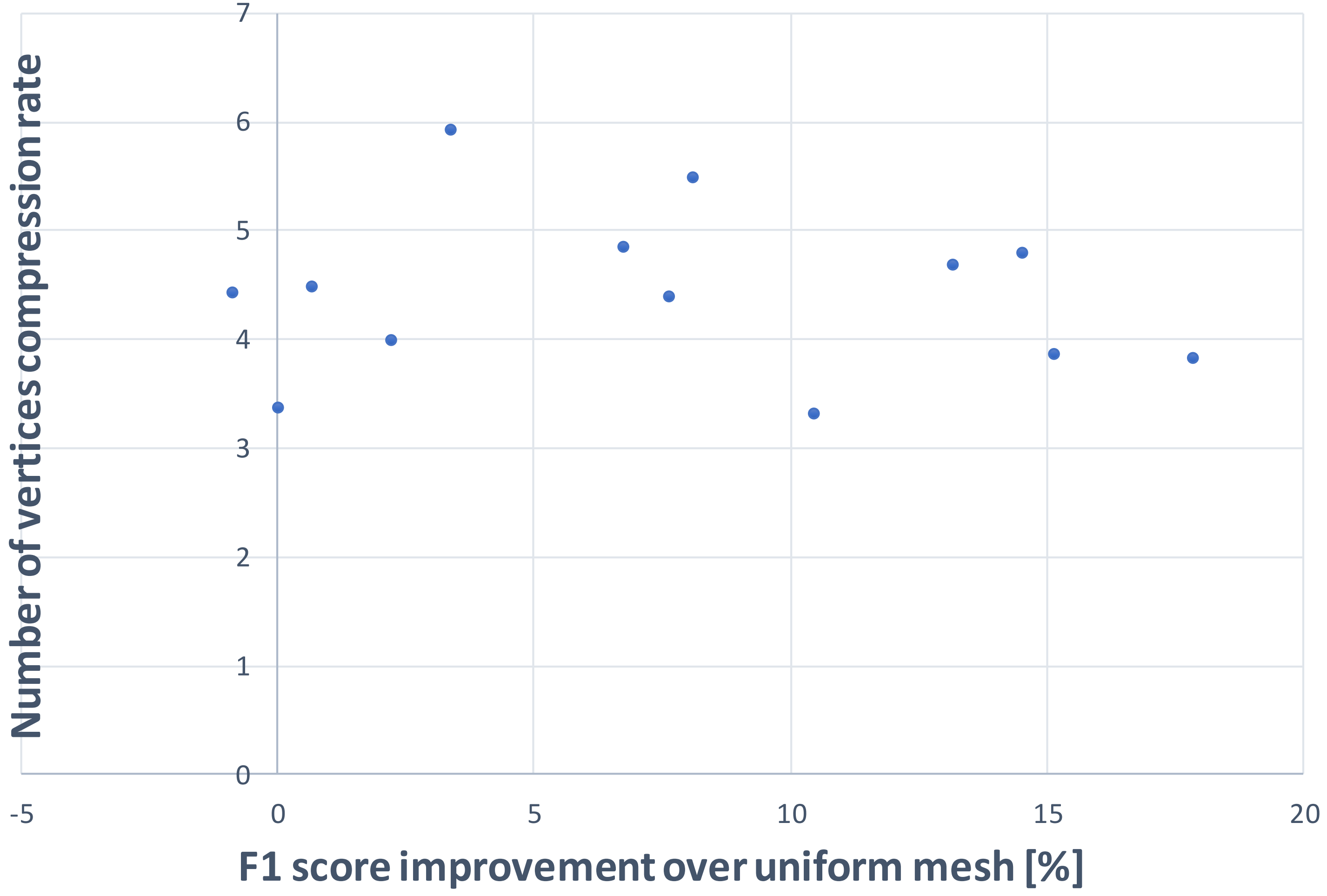
Repeat the pipeline with new initial mesh

- End to end, fully differentiable
- Encourages the generation of adaptive meshes

Full Mesh Generation Pipeline



Quantitative Results

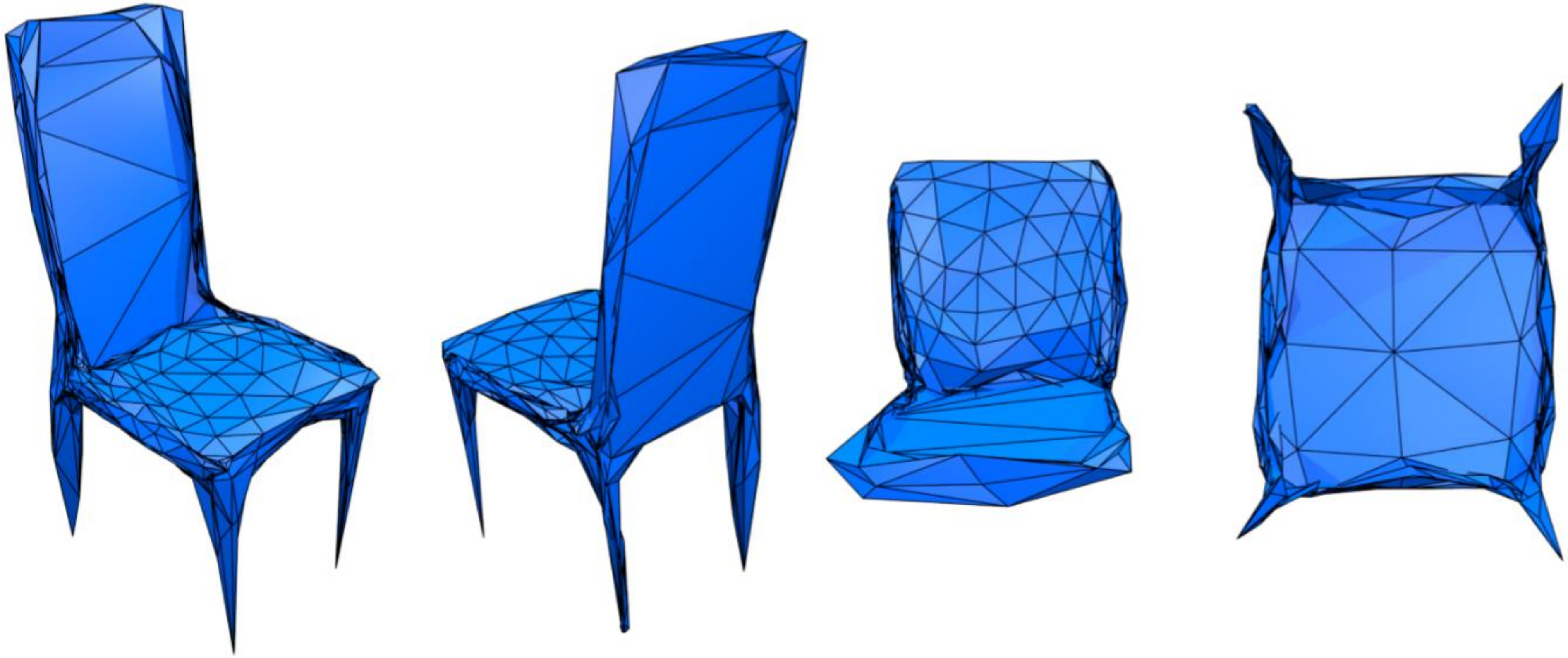


Quantitative Results

Ablation study:

Ours	GCN	Unif. Split.	No $\mathcal{L}_{latent}$	$\mathcal{L}_{\mathrm{VtP}}$	Pixel2Mesh
56.61	54.57	50.33	55.59	52.92	38.13

Qualitative Results



Qualitative Results

GEOMetric: Exploiting Geometric Structure for Graph-Encoded Objects

Visit our poster: 06:30 -- 09:00 PM @ Pacific Ballroom #145

Email us at: edward.smith@mail.mcgill.ca

Source code: https://github.com/EdwardSmith1884/GEOMetrics

Thank you for listening.