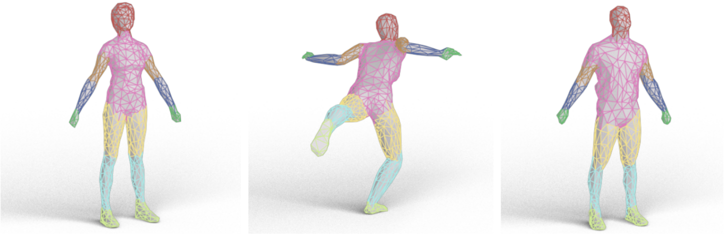
Geometric Deep Learning

Julie Digne

Master ID3D
LIRIS - CNRS
Équipe Origami

16/11/2023

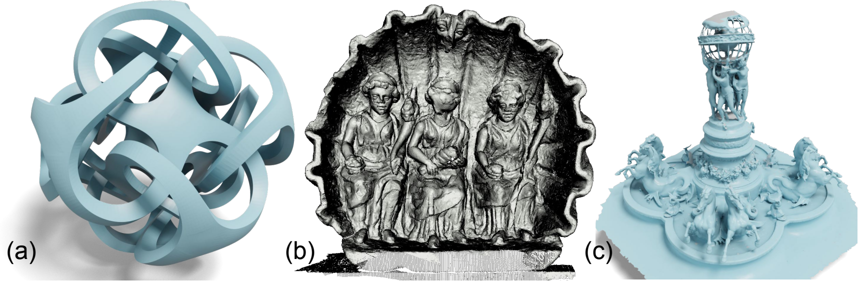
Teaser



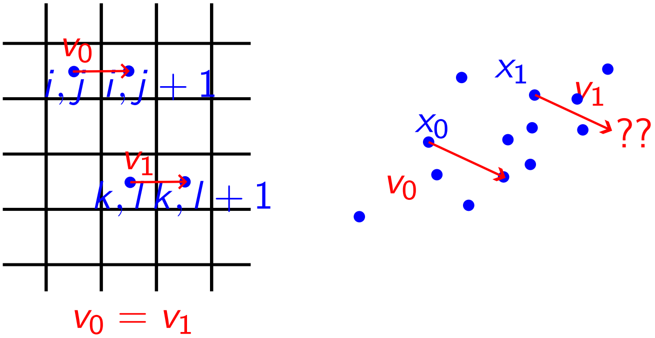
MeshCNN [Hanocka et al. 2019]

Outline

Image versus geometry

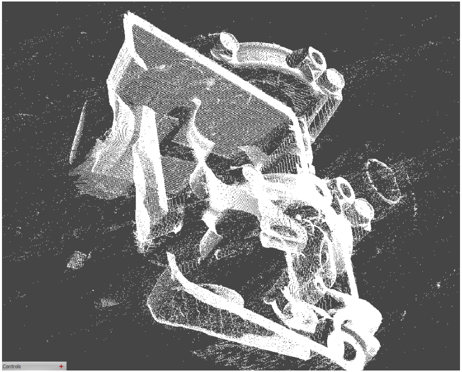


Geometric data



No grid structure.

Sampling issues



Irregular Sampling, occlusions when scanning

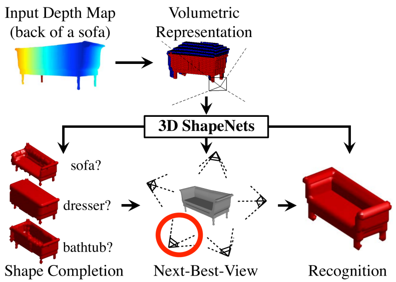
Geometric Deep Learning

- No image-like grid structure
- What is a good representation for working on geometric data?
- Various representations Meshes, Point sets...$\rightarrow$ **Networks adapted to this kind of data**

Outline

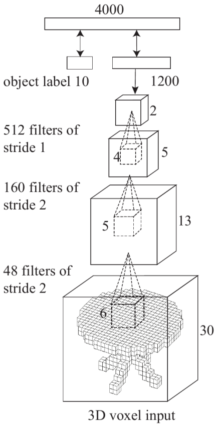
3D CNN

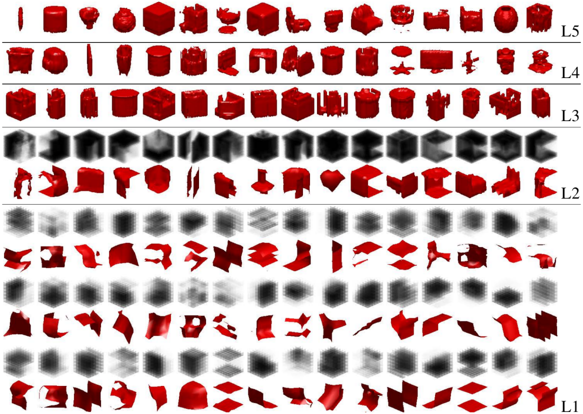
- *3D ShapeNets*
- Represents a shape as a probability distribution over a voxel grid.
- Learns the model distribution over voxels+classes.



[Wu et al. 2015]

3D CNN

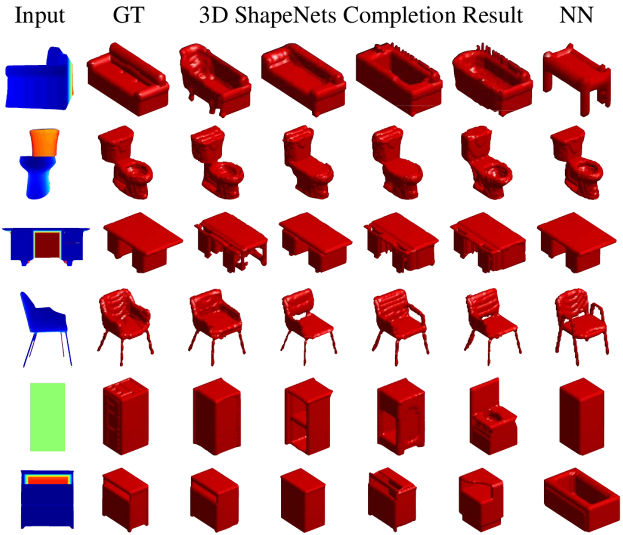




(b) Data-driven visualization: For each neuron, we average the top 100 training examples with

[Wu et al. 2015]

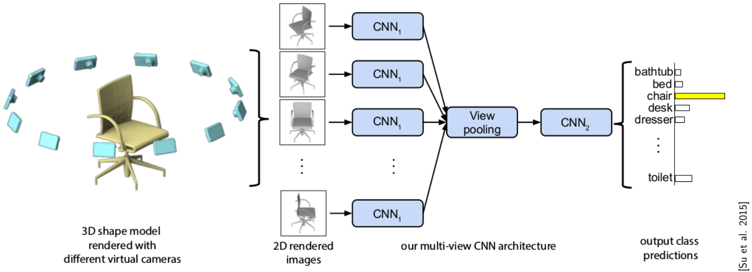
3D CNN - Shape completion



[Wu et al. 2015]

Issue: extremely low resolution: 24x24x24 (+padding)

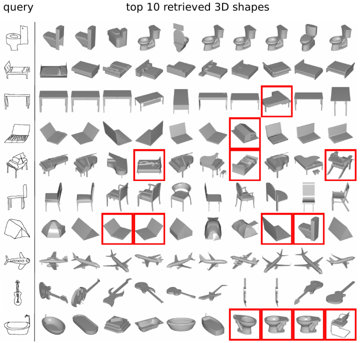
Multiview CNN [Su 2015]



Benefit from 2D convolution in a 3D-consistent manner.

Multiview CNN [Su 2015]

- Render a mesh from several viewpoints (up to 80)
- Process each image separately through a CNN



[Su et al. 2015]

Multiview aggregation

- CNN features (or SIFT features) used as a vector description, min distance between the view features
- **View-pooling: take the maximum feature values per pixel across all views.**

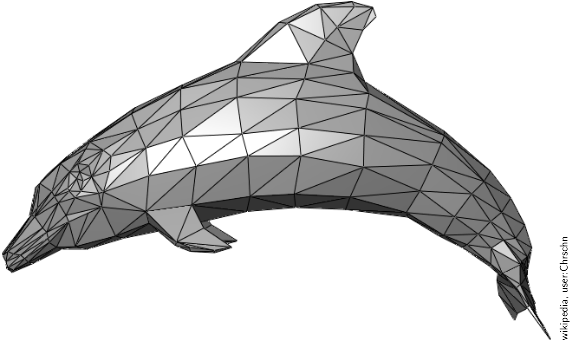
Multiview CNN [Su 2015]

Method	Training Config.			Test Config.	Classification (Accuracy)	Retrieval (mAP)
	Pre-train	Fine-tune	#Views	#Views		
(1) SPH [16]	-	-	-	-	68.2%	33.3%
(2) LFD [5]	-	-	-	-	75.5%	40.9%
(3) 3D ShapeNets [37]	ModelNet40	ModelNet40	-	-	77.3%	49.2%
(4) FV	-	ModelNet40	12	1	78.8%	37.5%
(5) FV, 12×	-	ModelNet40	12	12	84.8%	43.9%
(6) CNN	ImageNet1K	-	-	1	83.0%	44.1%
(7) CNN, f.t.	ImageNet1K	ModelNet40	12	1	85.1%	61.7%
(8) CNN, 12×	ImageNet1K	-	-	12	87.5%	49.6%
(9) CNN, f.t.,12×	ImageNet1K	ModelNet40	12	12	88.6%	62.8%
(10) MVCNN, 12×	ImageNet1K	-	-	12	88.1%	49.4%
(11) MVCNN, f.t., 12×	ImageNet1K	ModelNet40	12	12	89.9%	70.1%
(12) MVCNN, f.t.+metric, 12×	ImageNet1K	ModelNet40	12	12	89.5%	**80.2**%
(13) MVCNN, 80×	ImageNet1K	-	80	80	84.3%	36.8%
(14) MVCNN, f.t., 80×	ImageNet1K	ModelNet40	80	80	**90.1**%	70.4%
(15) MVCNN, f.t.+metric, 80×	ImageNet1K	ModelNet40	80	80	**90.1**%	79.5%

* f.t.=fine-tuning, metric=low-rank Mahalanobis metric learning

[Su et al. 2015]

Meshes



- When the data is represented as a mesh: there is some structure even if irregular!
- Mesh can be seen as a graph
- Graph CNN

Meshes vs graphs

Meshes are very special types of graphs, they define a manifold surface.

Graph Neural Networks [Gori et al. 2005, Scarselli et al. 2005]

- Message passing between neighboring nodes
- Each nodes aggregates the messages and updates them
- Per node task: process the resulting per-node feature vectors
- Per graph task: aggregates the per-node feature vectors

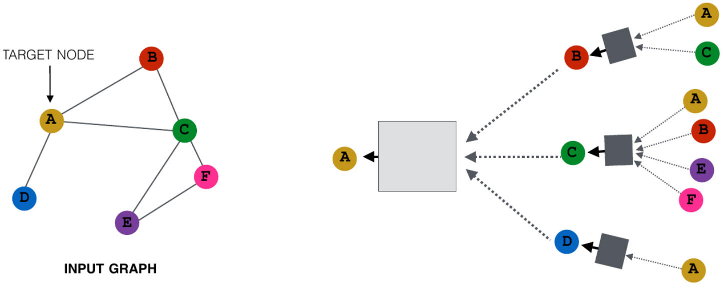


Image by Jure Leskovec (Stanford)

Aggregation function

- For per-node aggregation: should be independent on the order (permutation invariance)
- In per-node tasks: the resulting vector should be permutation equivariant

Aggregation function

- For per-node aggregation: should be independent on the order (permutation invariance)
- In per-node tasks: the resulting vector should be permutation equivariant

Permutation-invariant functions

average, max, min, sum

Aggregation function

- For per-node aggregation: should be independent on the order (permutation invariance)
- In per-node tasks: the resulting vector should be permutation equivariant

Permutation-invariant functions

average, max, min, sum

Many GNN variants

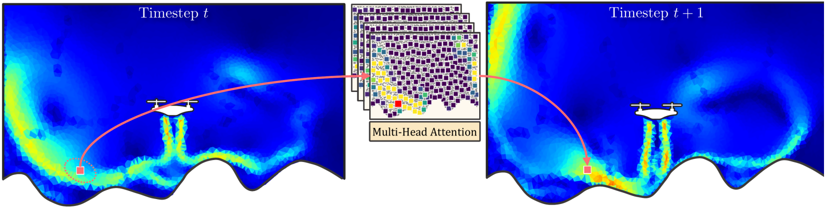
Features can also be on edges (dual graph), or on both edges and vertices. Graph CNN: convolution by a kernel $g_\theta = diag(\theta)$, U matrix of eienvectors of the normalized graph laplacian.

$$g_\theta \star x = U g_\theta U^T x$$

Graph Neural Networks – new version

Graph transformers

Transformer on graphs, large receptive field.

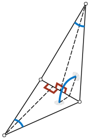


[Janny, Nadri, Digne, Thome, Wolf 2023]

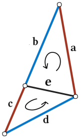
- Used in many machine learning-based physics simulation.

MeshCNN [Hanocka et al. 2019]

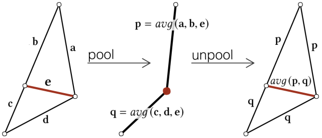
- Defines convolution and pooling layers on mesh edges.
- Meshes are assumed manifold, possibly with boundary vertices.
- Pooling prioritized by smallest edge feature.



Input Edge features



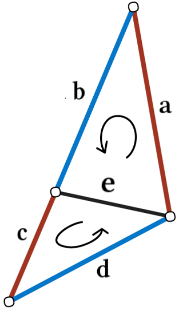
Convolution operation



Pooling and unpooling

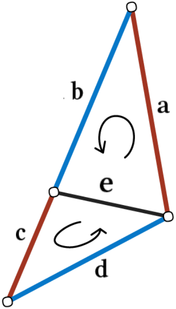
[Hanoka et al.]

MeshCNN - Convolution on edges



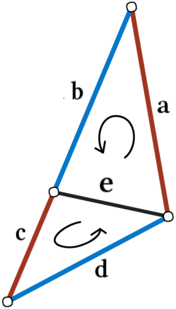
- Convolution: $e * k_0 + \sum_{i=1}^{4} k_i e_i$

MeshCNN - Convolution on edges



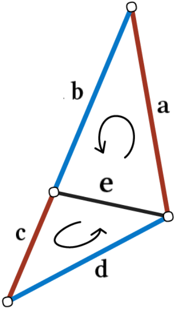
- Convolution: $e * k_0 + \sum_{i=1}^{4} k_i e_i$
- Ambiguity: $e * k_0 + a * k_1 + b * k_2 + c * k_3 + d * k_4$ or $e * k_0 + c * k_1 + d * k_2 + a * k_3 + b * k_4$

MeshCNN - Convolution on edges



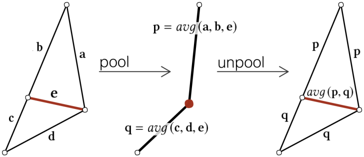
- Convolution: $e * k_0 + \sum_{i=1}^{4} k_i e_i$
- Ambiguity: $e * k_0 + a * k_1 + b * k_2 + c * k_3 + d * k_4$ or $e * k_0 + c * k_1 + d * k_2 + a * k_3 + b * k_4$
- Solution: work with $(|a - c|, a + c, |d - b|, d + b)$

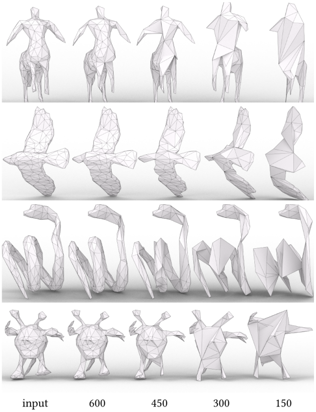
MeshCNN - Convolution on edges



- Convolution: $e * k_0 + \sum_{i=1}^{4} k_i e_i$
- Ambiguity: $e * k_0 + a * k_1 + b * k_2 + c * k_3 + d * k_4$ or $e * k_0 + c * k_1 + d * k_2 + a * k_3 + b * k_4$
- Solution: work with $(|a - c|, a + c, |d - b|, d + b)$
- Then usual 2d convolution on these "fake edge features"

MeshCNN - Pooling on edges



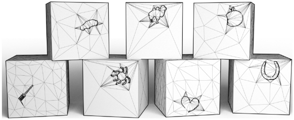


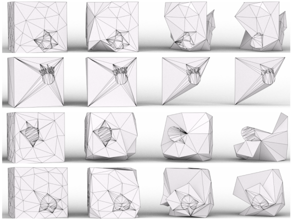
- Not all edges can collapse: prevent non-manifold faces creating edge collapses.
- Control the target mesh resolution by setting the targer number of edges.
- Store the history of pooling $\rightarrow$ can reinstore the original mesh topology.

t al. 2019]

MeshCNN: application to mesh classification

- Add a global pooling layer and linear layers, after several meshcnn layers.





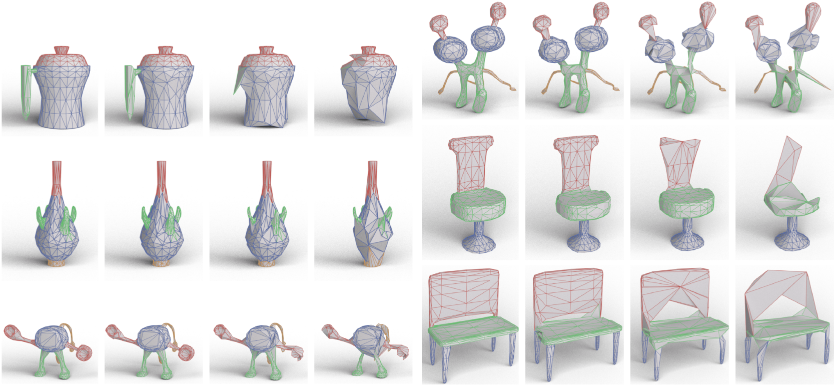
Cube Engraving Classification

method	input res	test acc
MeshCNN	750	**92.16%**
PointNet++	4096	64.26%

[Hanocka et al. 2019]

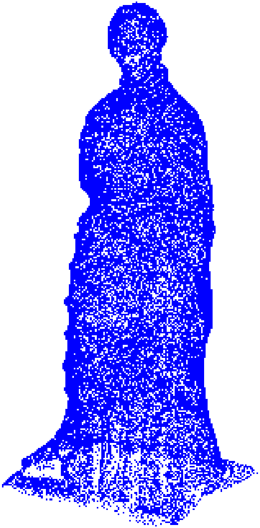
MeshCNN: application to mesh segmentation

- Only meshcnn layers.



[Hanocka et al. 2019]

Point sets

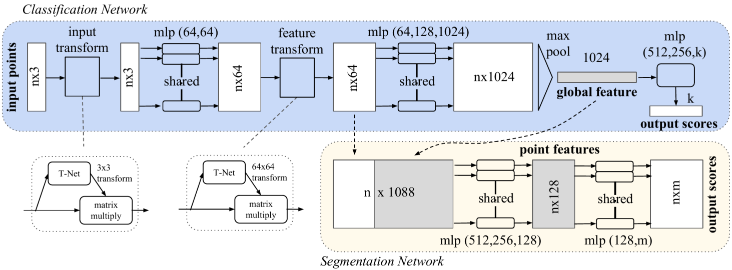


- No structure anymore
- Missing data
- Various number of points, point ordering can change.

PointNet [Qi 2017]

Principle

Affine transform per point followed by permutation invariant pooling on channels



[Qi et al. 2017]

PointNet - An approximation theorem

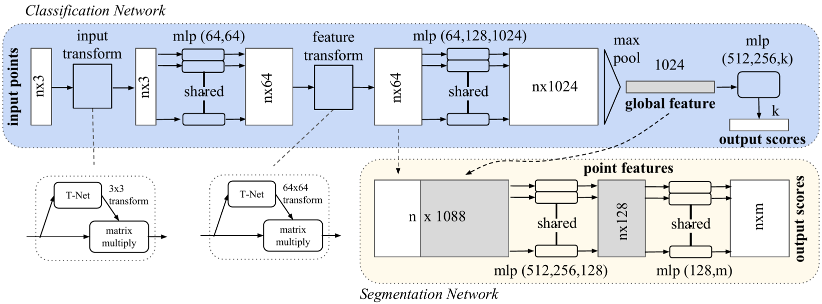
Theorem 1. *Suppose $f : \mathcal{X} \rightarrow \mathbb{R}$ is a continuous set function w.r.t Hausdorff distance $d_H(\cdot, \cdot)$. $\forall \epsilon > 0$, $\exists$ a continuous function h and a symmetric function $g(x_1, \dots, x_n) = \gamma \circ MAX$, such that for any $S \in \mathcal{X}$,*

$$\left| f(S) - \gamma \left(\underset{x_i \in S}{MAX} \{h(x_i)\} \right) \right| < \epsilon$$

where $x_1, \dots, x_n$ is the full list of elements in S ordered arbitrarily, γ is a continuous function, and MAX is a vector max operator that takes n vectors as input and returns a new vector of the element-wise maximum.

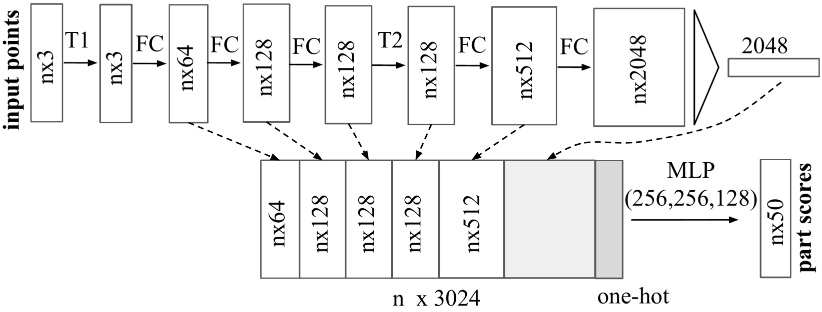
- Proof derives directly from the universal approximation theorem.

PointNet - Results



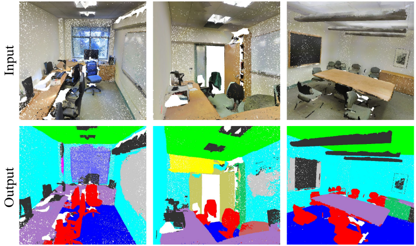
[Qi et al. 2017]

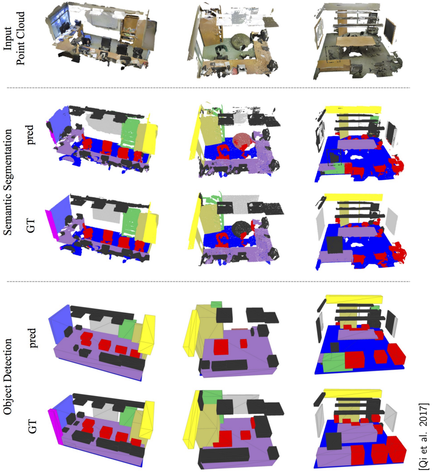
PointNet - Results



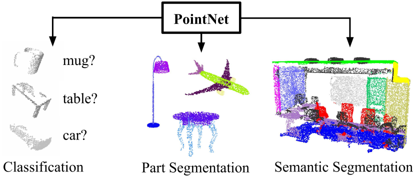
[Qi et al. 2017]

PointNet - Results





PointNet - Results



	#params	FLOPs/sample
PointNet (vanilla)	0.8M	148M
PointNet	3.5M	440M
Subvolume [18]	16.6M	3633M
MVCNN [23]	60.0M	62057M

[Qi et al. 2017]

Issues

Looses locality. Improved in PointNet++ (also in 2017).

Light Networks

- Deep networks are expansive (large computation time and environmental cost)
- PointNet is rather light
- Combine pointnet + light 2D network to get competitive results for RGBD segmentation.

Methods	InputType	GT	NbParam	2D backbone	mIoU
CMX* [29]	RGB + Depth (HHA)	2D	66 M	SegFormer-B2	51.3
RFBNet [36]	RGB + Depth (HHA)	2D	No info	ResNet-50	62.6
Ours (LPointNet + U-Net34)	RGB + Point cloud from Depth	2D	26 M	ResNet-34	**63.2**
SSMA [37]	RGB + Depth (HHA)	2D	56 M	AdaptNet++	66.3
ShapeConv [28]	RGB + Depth (HHA)	2D	58 M	Deeplabv3+	66.6
3D-to-2D distil [30]	RGB + Point cloud	2D	66M	ResNet-50	58.2
Ours (KPConv + U-Net34)	RGB + Point cloud	2D	49 M	ResNet-34	**63.8**
BPNet* [2]	RGB + Point cloud	2D/3D	96 M	ResNet-34	64.4
Ours (LPointNet + U-Net34)	RGB + Point cloud	2D	**26 M**	ResNet-34	**66.1**
VirtualMVFusion [25] (single view)	RGB + Normals + Coordinates	3D	No info	xcpetion65	67.0
Ours (LPointNet + SegFormer-B2)	RGB + Point cloud	2D	30 M	SegFormer-B2	**69.0**

[Pradelle, Chaine, Wendland, Digne 2023]

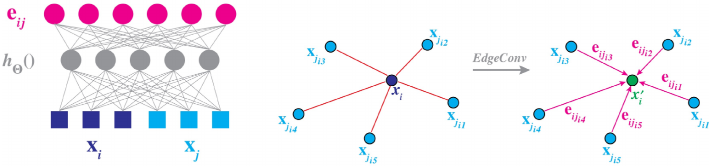
Dynamic Graph CNN [Wang 2019]

- Builds a k-nearest neighbors graph
- Defines an edge convolution

Idea

Recompute the nearest neighbor graph in the feature space after each layer.

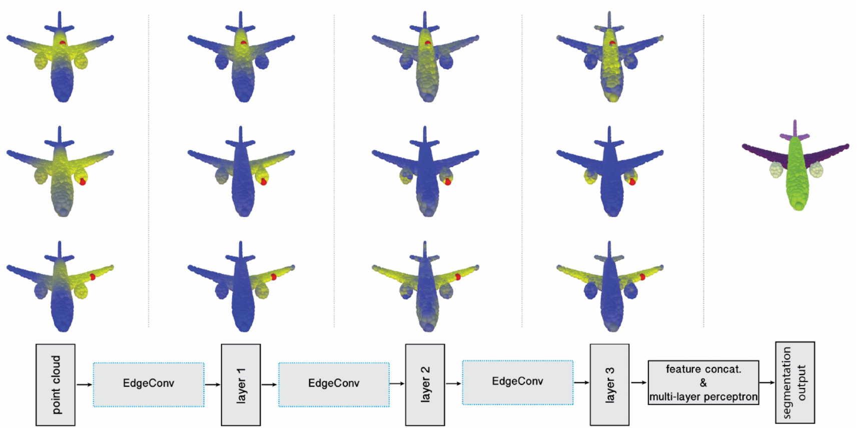
DGCNN - Edge convolution



[Wang et al. 2019]

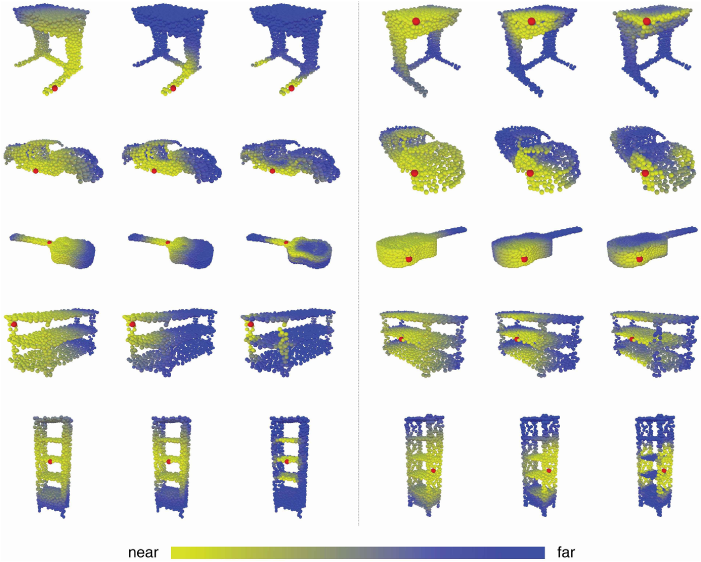
- Compute an edge feature using an MLP on the channels of the end vertices
- Aggregate the edge features by permutation invariant pooling on each vertex

DGCNN - Architecture



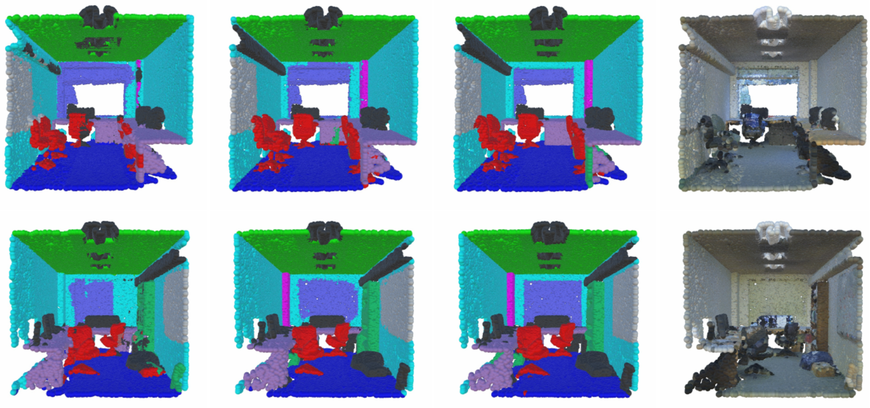
[Wang et al. 2019]

DGCNN - feature distance



[Wang et al. 2019]

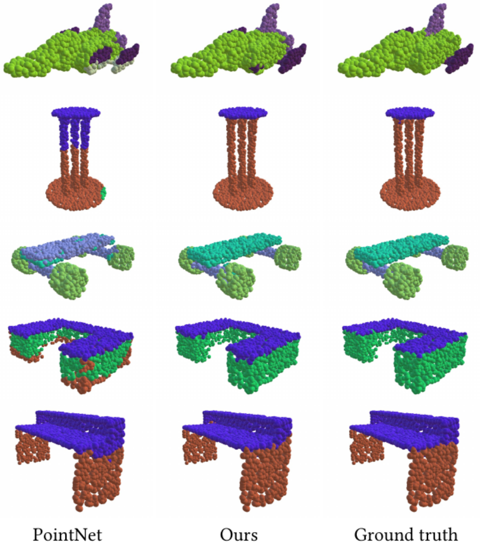
DGCNN - Results



PointNet Ours Ground truth Real color

[Wang et al. 2019]

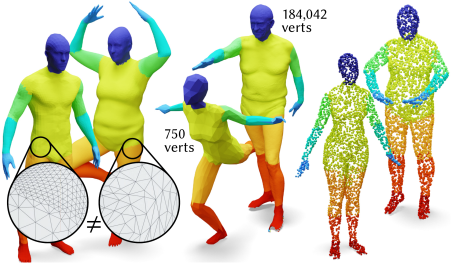
DGCNN - Results



[Wang et al. 2019]

Diffusion is all you Need [Sharp 2022]

- Representation agnostic model, based on diffusion on the shape

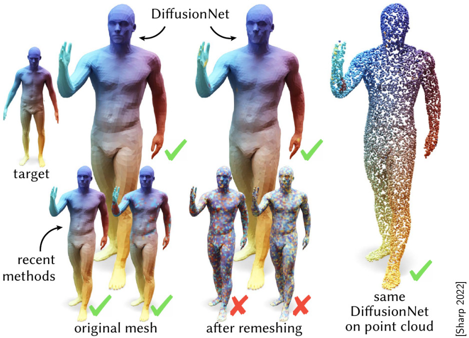


[Sharp 2022]

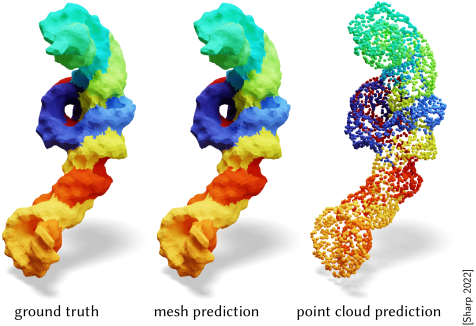
Diffusion is all you Need

- $u \in \mathbb{R}^V$ feature + obtained by pointwise MLP
- $\frac{\partial x}{\partial t} = \Delta x(t)$
- Diffusion layer $H_t(u_0) = exp(t\Delta)u_0$, use the Laplacian eigenbasis to reduce computation load
- To get non radially symmetric filters: add local gradient operators.

Diffusion is all you Need - Results



Diffusion is all you Need - Results

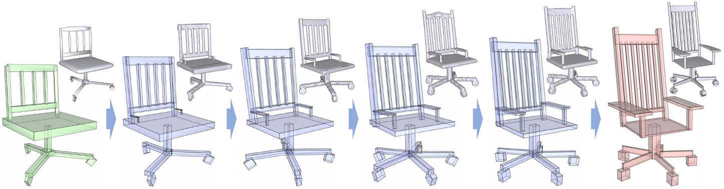


Outline

But: only for *static* geometry

How do we cope with generative tasks

An example for generating shapes [GRASS, Li et al. 2017]

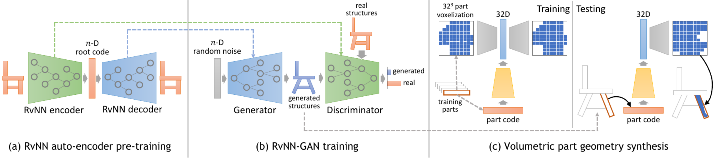


[Li et al. 2017]

- Input data: set of shapes with a semantic segmentation into parts.

Algorithm

- Step 1: Learn a code representing an arrangement of boxes.
- Step 2: Train a GAN for generating a new structure
- Step 3: Use voxelization in each box to synthesize the local geometry.



[Li et al. 2017]

Step 1: Learn a code

Key idea

Shape components are commonly arranged or perceived to be arranged hierarchically. Goal of the code: encode this hierarchy of parts

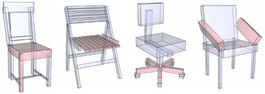
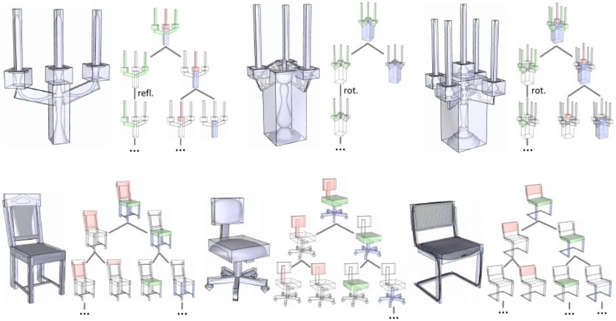


Fig. 3. Merging criteria used by our model demonstrated with 3D shapes represented by part bounding boxes (relevant parts highlighted in red). From left: (a) two adjacent parts, (b) translational symmetry, (c) rotational symmetry, and (d) reflective symmetry.

[Li et al. 2017]

- Recursive auto-encoder for binary trees: encode the structure into a code; decode and compare the recovered structure.
- Recursively merge parts that are either adjacent or symmetric (rotational, translational, reflectional)
- Training: generate plausible hierarchies for each shape (sample the space of plausible part groupings)
- Adjacency and Symmetry encoder/decoder (transform a code into another encodes the symmetry and the generator)
- Additionally: Box encoder/Node classifier

Learned hierarchies

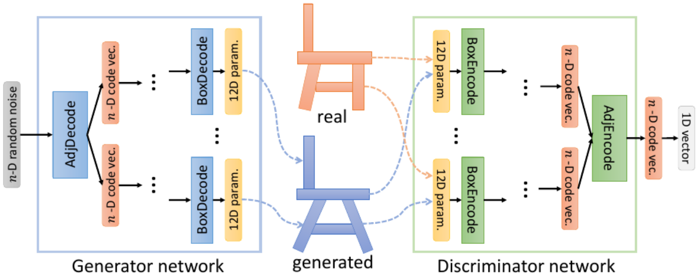


[Li et al. 2017]

In a nutshell

Transform a binary tree into a meaningful hierarchy while minimizing the loss (sum of bounding boxes distances)

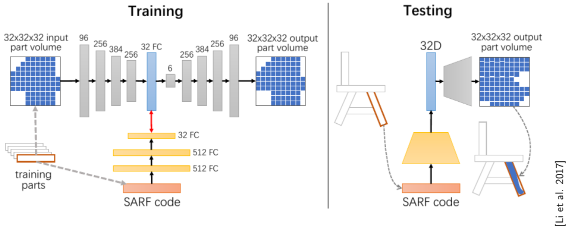
Etape 2: encoder-decoder model for generation



[Li et al. 2017]

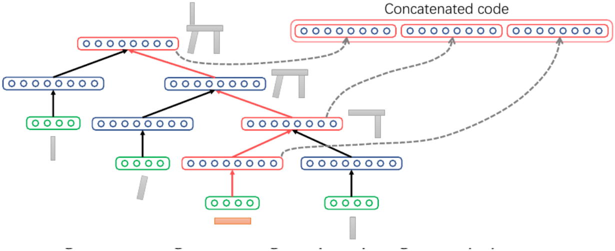
- Idea: adversarial training: the generator tries to fool the discriminator which in turns tries to detect generated pairs.
- Prior structure for the input to the generator (sample from the set of input and generated output hierarchies for the auto-encoder + other tricks)

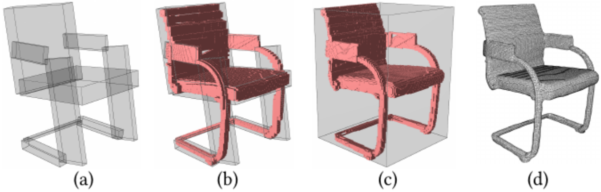
Etape 3: geometry synthesis



- Goal: synthesize a coherent voxel grid for each bounding box representing the fine-grained geometry
- Take into account both the geometry of the bouding box and its context

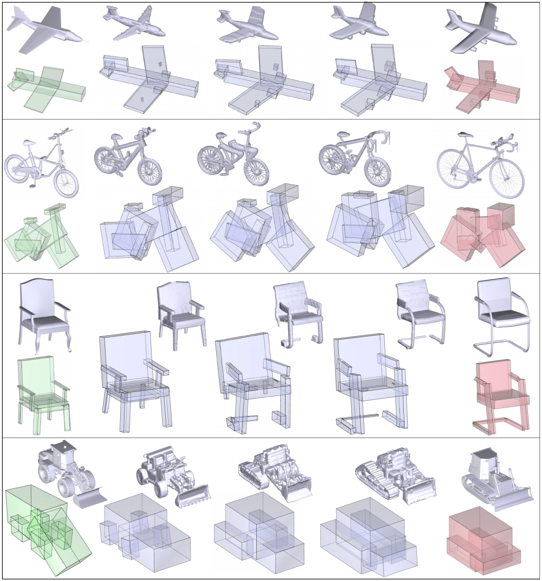
Contextual description and final synthesis





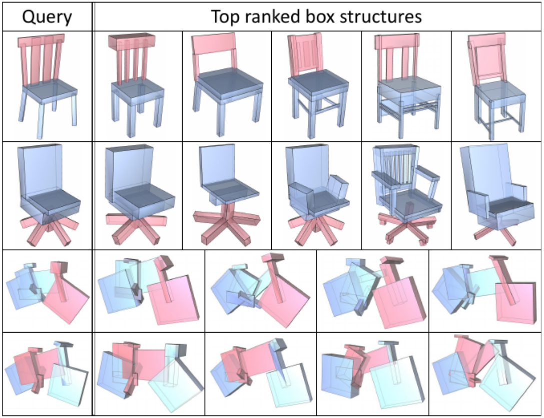
[Li et al. 2017]

Application: interpolation



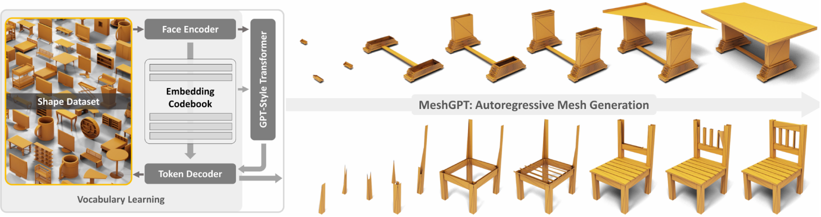
[Li et al. 2017]

Application: shape query



[Li et al. 2017]

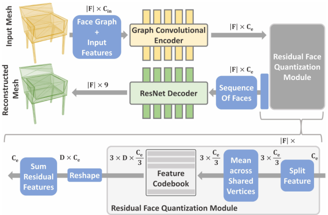
MeshGPT [Siddiqi et al. 2023] - released 3 days ago!



[Siddiqi et al. 2023]

- Following text generation idea: generate a mesh as a sequence of triangles

MeshGPT - Principle



[Siddiqi et al. 2023]

- Learns a vocabulary of latent representations of faces
- Uses these latent representations as tokens
- GPT-like transformer: predicts next token from previous tokens auto-regressively.
- 1D Resnet decodes the latent representation sequences into triangles

MeshGPT - Architecture details

- Graph CNN encoder on the graph of faces (each face = a node) learns a latent per face representation, input features: vertex coordinates (9-dimensional).
- SAGE convolution layer: samples neighborhood and aggregates features from it. For a mesh of N faces:

$$Z = (z_1, \cdots, z_N)$$

- Residual Vector Quantization: quantization on a primary codebook, residuals quantized on a secondary codebook... Yields a codebook and D codes per face (with additional tricks)

$$T = (t_1, \cdots, t_N);\ t_i = t_i^j \text{ index of an embedding in the codebook.}$$

- Decoder (1d resnet) G decodes the token into 9 coordinates.
- Codebook and graph encoder given to the transformer using T as a sequence.

Result

Resuls is a triangle soup: needs post-processing to turn it into a watertight mesh

MeshGPT - Results

[Siddiqi et al. 2023]

MeshGPT - Results

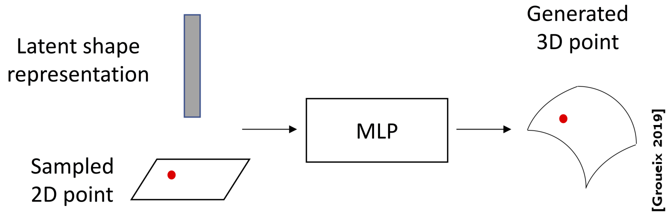
[Siddiqi et al. 2023]

Outline

Machine learning based surface reconstruction

- Needs a differentiable pipeline
- Challenge: intrinsically a combinatorial problem...
- Not necessarily example-based: surface reconstruction can be done per shape.

AtlasNet [Groueix 2019]



- Some definitions:
  - A manifold surface $\mathcal{S}$ in $\mathbb{R}^3$ is topological set such that each point has a neighborhood which is homeomorphic to an open disk of *mathbbR*^{2}.
  - Local map (or chart):s a homeomorphism φ from an open subset U of $\mathcal{S}$ to an open subset of $\mathbb{R}^2$.
  - Atlas:a indexed family of local charts (U_i, ϕ_i) from U_i to open subsets of $\mathbb{R}^2$; such that the U_is cover $\mathcal{S}$.

Parameterization

This is the base for surface parameterization problems in geometry processing: Try to unwrap a surface onto a planar patch (usually a square).

AtlasNet [Groueix 2019]

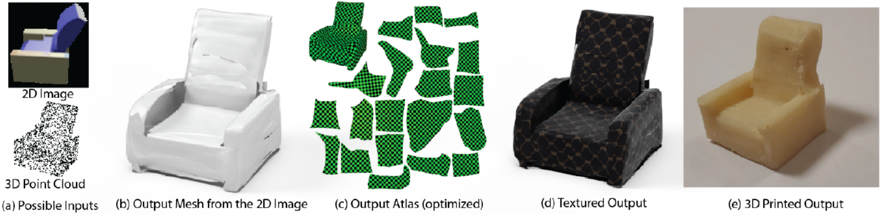
- Model the local maps as affine maps, they can be inverted if they are full rank.
- A ReLU-based MLP computes a piecewise affine map (full rank). This is due to ReLU activation.
- Start with N patches and compute their deformation onto the surface (*Papier mâché*). Deformed patches may overlap.

AtlasNet for surface reconstruction

- Start with a latent representation x of a shape
- For a set of points $\mathcal{A}$ of points sampled in $[0,1]^2$, we optimize the weights θ_i of N functions (MLP) f_{θ_i}
- Sample a set $\mathcal{S}_d$ of M points on the surface $\mathcal{S}$
- Chamfer Loss

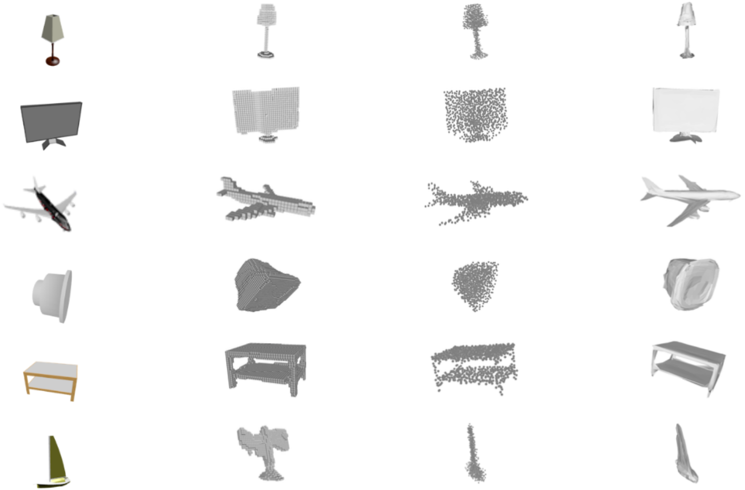
$$\sum_{p\in\mathcal{A}}\sum_{i=1}^{N}\min_{q\in\mathcal{S}_D}\|f_{\theta_i}(p,x)-q\|_2^2+\sum_{q\in\mathcal{S}_d}\min_{i=1\cdots N}\min_{p\in\mathcal{A}}\|f_{\theta_i}(p,x)-q\|^2$$

Result



[Groueix et al. 2019]

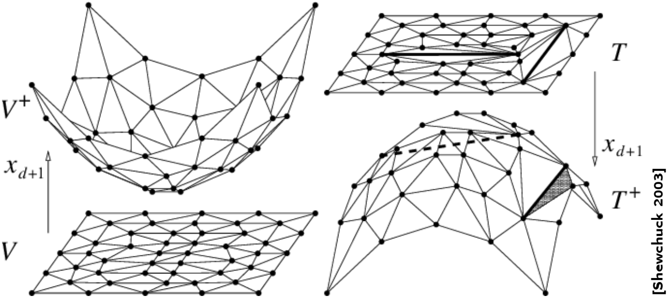
Results: reconstruction from single view



(a) input (b) 3D-R2N2 (c) PSG (d) Ours

[Groueix et al. 2019]

Differentiable Surface Reconstruction [Rakotosaona 2021]



- A set of points $v_j \in \mathbb{R}^d$ with weights w_j
- *Weighted Delaunay Triangulation:* projected lower envelop of points $(v_j, \|v_j\|^2 - w_j) \in \mathbb{R}^{d+1}$
- Any 2D ($d = 2$) triangulation can be obtained as a perturbation of a 2d Weighted Delaunay Triangulation.

Differentiable ~~weighted~~ Delaunay triangulation in 2D

- All possible triangles with vertices in V are given an inclusion score e_i.
- defs: c_i circumcenter of triangle $i = \{j, k, l\}$, $a_{i|j}$ reduced Voronoi cell of vertex j onto triangle i. Then

$$e_i = \begin{cases} 1 & \text{if } c_i \in a_{x|i} \ \forall x \in \{j, k, l\} \\ 0 & \text{otherwise} \end{cases}$$

- *Continuous inclusion score*

$$s_{i|j} = \sigma(\alpha d(c_i, a_{j|i})) \ (\sigma \text{ sigmoid})$$

$$s_i = \frac{1}{3}(s_{i|j} + s_{i|k} + s_{i|l})$$

Differentiable weighted Delaunay triangulation in 2D

Weighted Voronoi cell a^w

Intersection of half planes $H_{j \leq k} = \{x \in \mathbb{R}^2 | \|x - v_j\|^2 - w_j \leq \|x - v_k\|^2 - w_k\}$

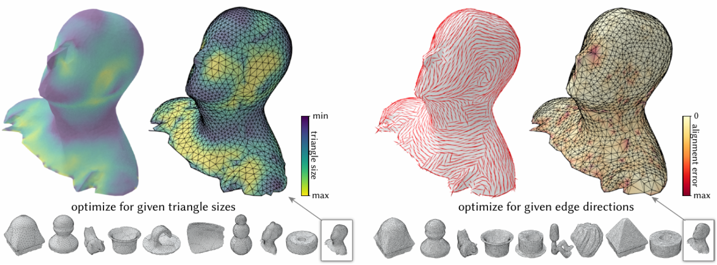
- redefine: c_i weighted circumcenter of triangle $i = \{j, k, l\}$, $a_{i|j}$ reduced weighted Voronoi cell of vertex j onto triangle i.
- Same expression for the continuous inclusion score

$$s_{i|j} = \sigma(\alpha d(c_i, a^w_{j|i})) \text{ } (\sigma \text{ sigmoid})$$

$$s_i = \frac{1}{3}(s_{i|j} + s_{i|k} + s_{i|l})$$

Turning 3D triangulation problems into 2d triangulation problems

- Segment 3D shapes into *developable sets* by Least Squares Conformal Maps [Lévy 2008].
- Differentiable 2D meshing on each of the sets with boundary constraints.



from [Rakotosaona 2021]

Losses

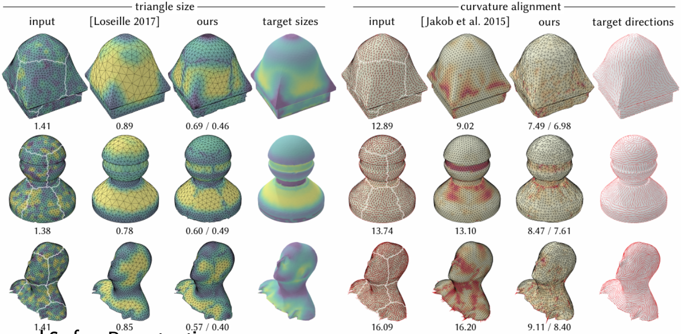
- Area prescribing loss (A: function on the surface):

$$\mathcal{L}_{area} = \frac{1}{\sum_{i,j} s_{i|j}} \sum_{i,j} (\frac{1}{2} \|(v_j - v_k) \times (v_l - v_k)\| - A(v_j))$$

- Boundary preservation loss:

$$\mathcal{L}_b(V, \mathcal{P}) = \frac{1}{|V|} \sum_j \exp(\varepsilon - \min(\varepsilon, (v_j - b_j) \cdot n_j^b))$$

- Other possible losses: angle loss, curvature alignment loss.



from [Rakotosaona 2021]

Conclusion

- Very small overview of geometric deep learning
- In particular, it's missing the nice definitions of equivariant convolutions or methods based on the bundle Laplacian.
- Missing also implicit surfaces: next time