Computer Visions
Generative Machine Learning Tools

MAS.S68 F’19
Roy Shilkrot, Fluid Interfaces

Class 6: 3D

Today

Catch up

- HW 3+4 update

Content

- Representing 3D models
- ML for 3D data
- Generative ML for 3D

Application

- Sketch to 3D, Photo to 3D
- 3D Latent space learning
- 3D GANs

Representing 3D Models

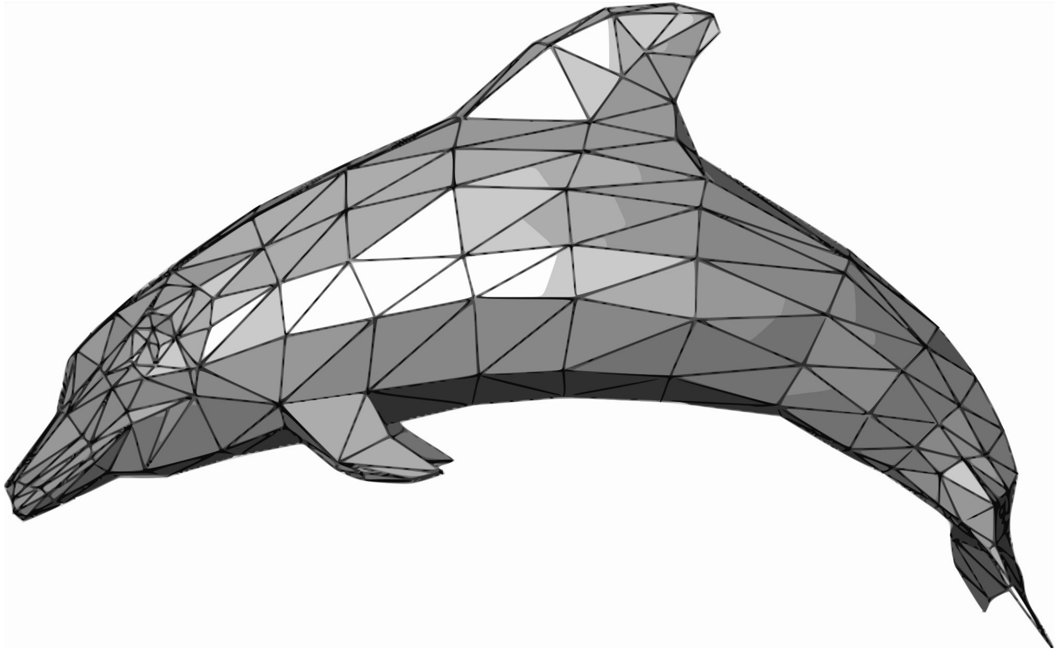
Mesh

Graph

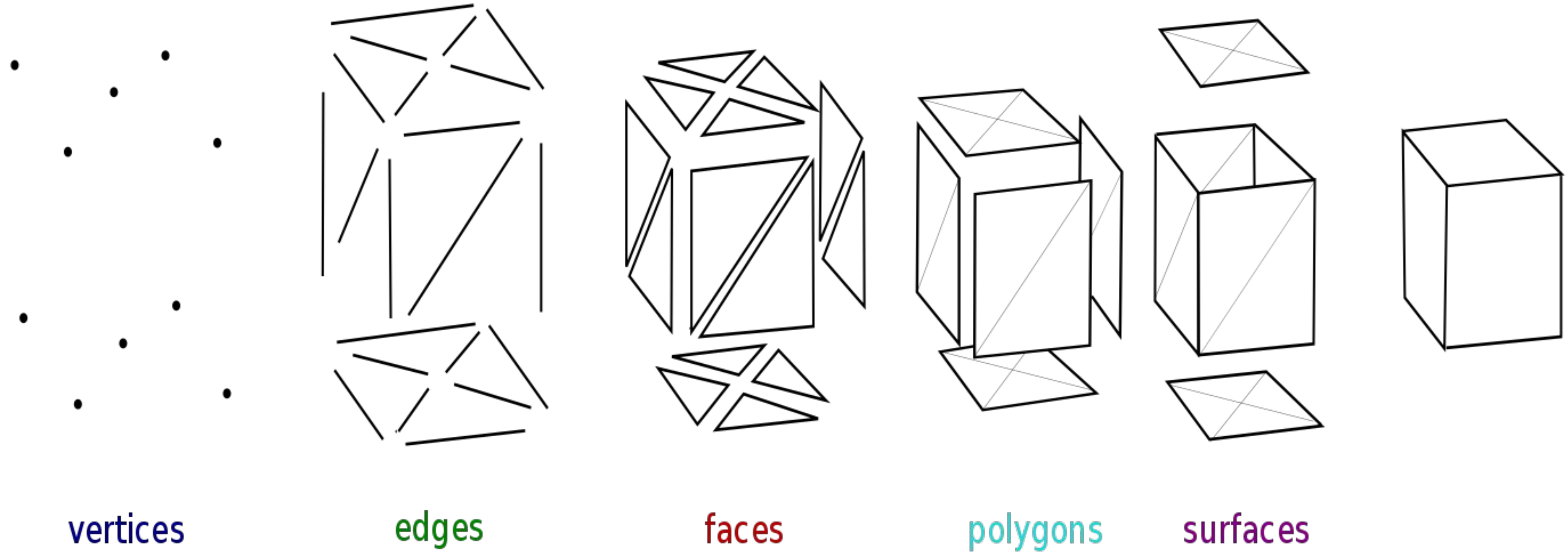
Volume

Parametric Design

Generative / Procedural Design

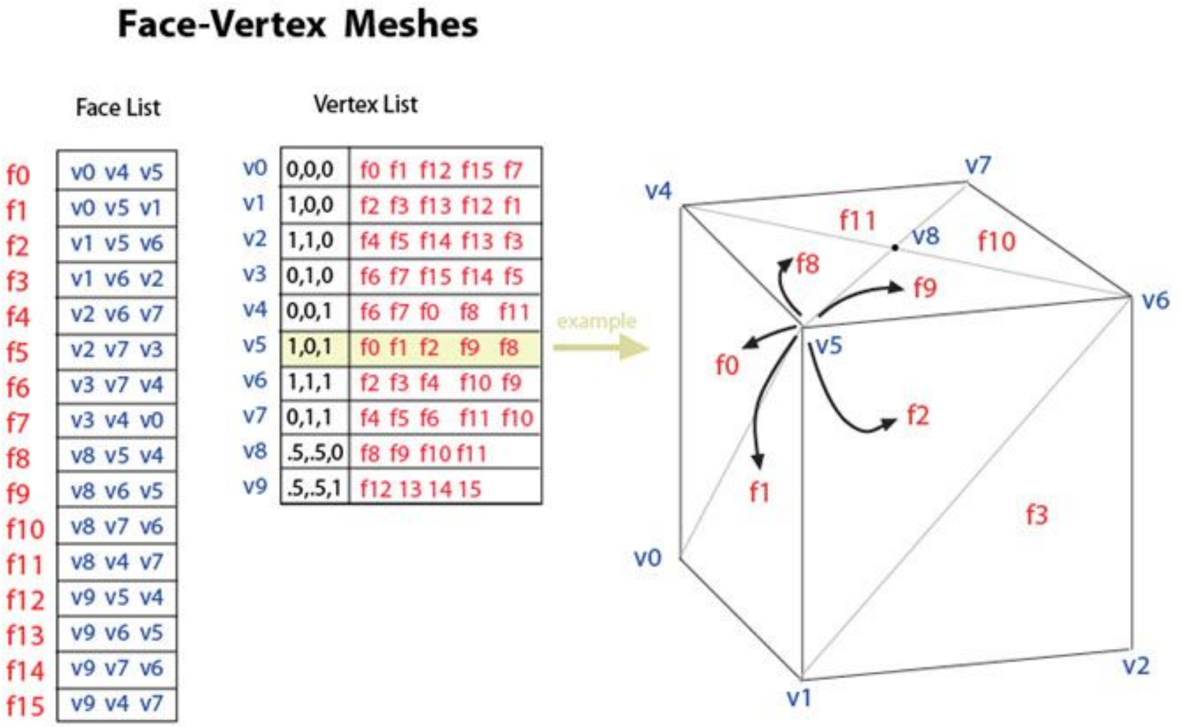


3D Model as Mesh

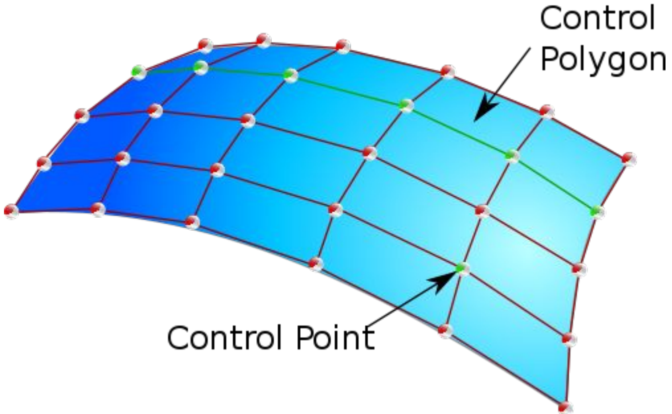


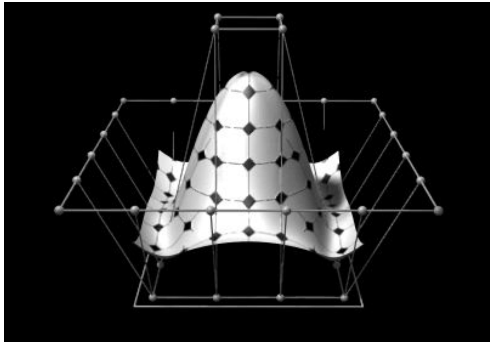
3D Model as Mesh

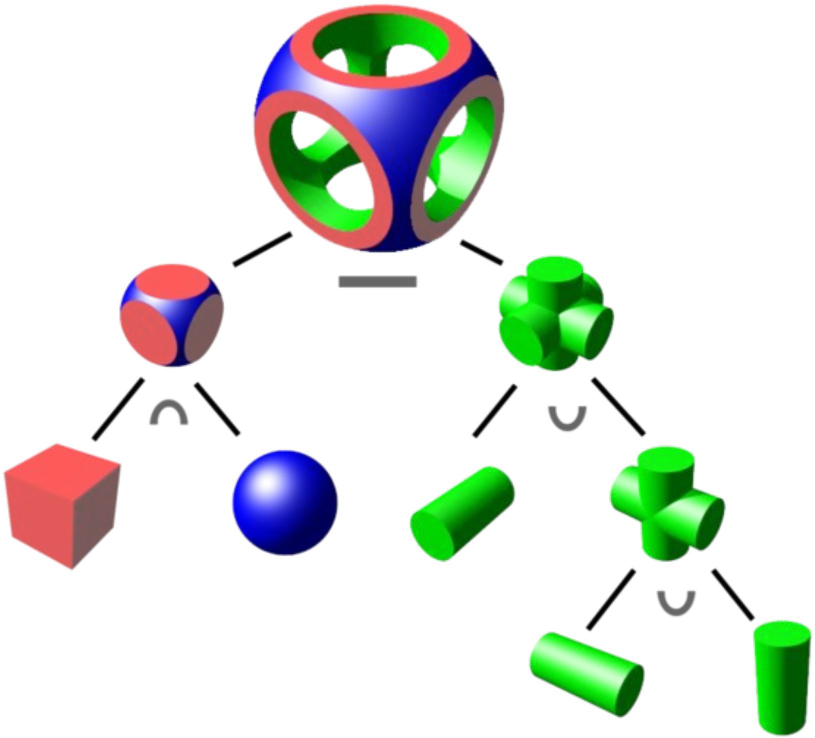
```
ply
format ascii 1.0
element vertex 867
property float32 x
property float32 y
property float32 z
element face 1704
property list uint8 int32 vertex_indices
end_header
0.00472708 0.0012 -0.000833515
0.0048 0.0012 0
0 0 0
0.00451052 0.0012 -0.0016417
0.00415692 0.0012 -0.00240001
0.00367701 0.0012 -0.00308539
0.00308537 0.0012 -0.00367702
0.00239999 0.0012 -0.00415693
0.00164168 0.0012 -0.00451053
3 0 1 2
3 3 0 2
3 4 3 2
3 5 4 2
3 6 5 2
3 7 6 2
3 8 7 2
3 9 8 2
3 10 9 2
```



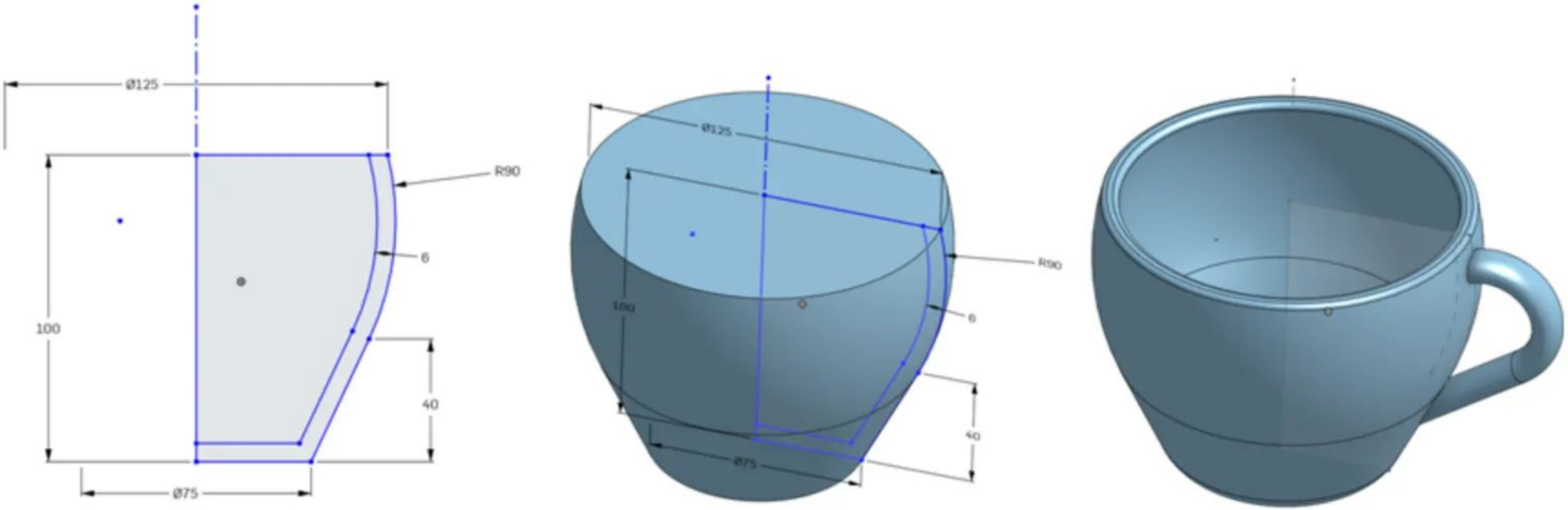
3D Model as Solid





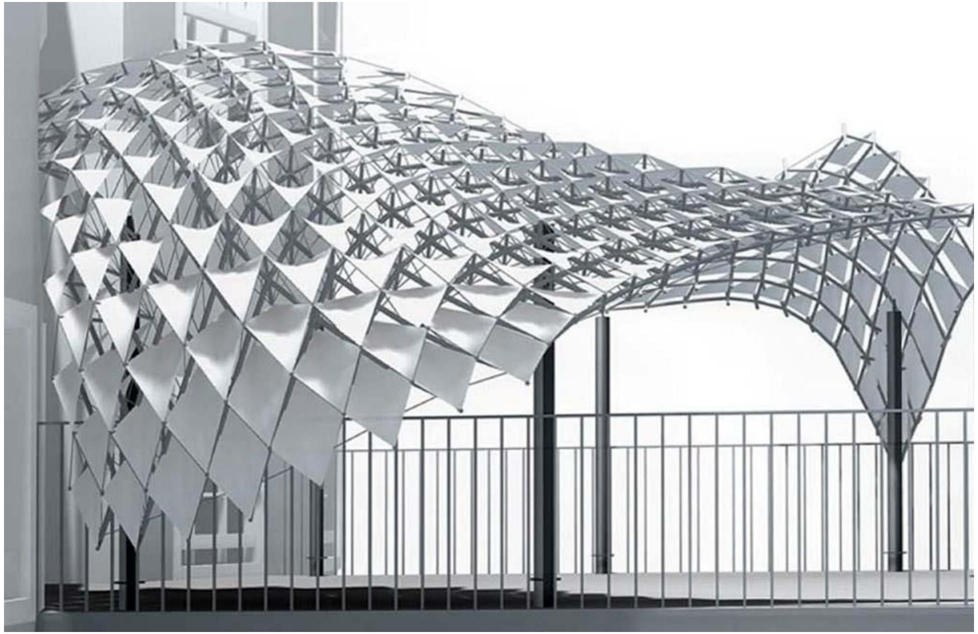


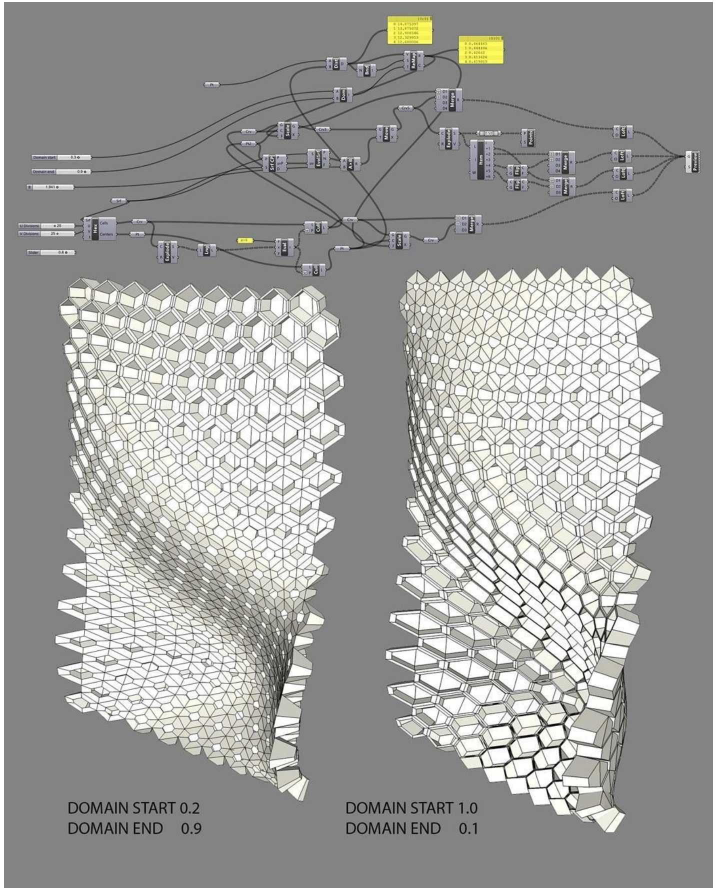
Parametric 3D Modeling



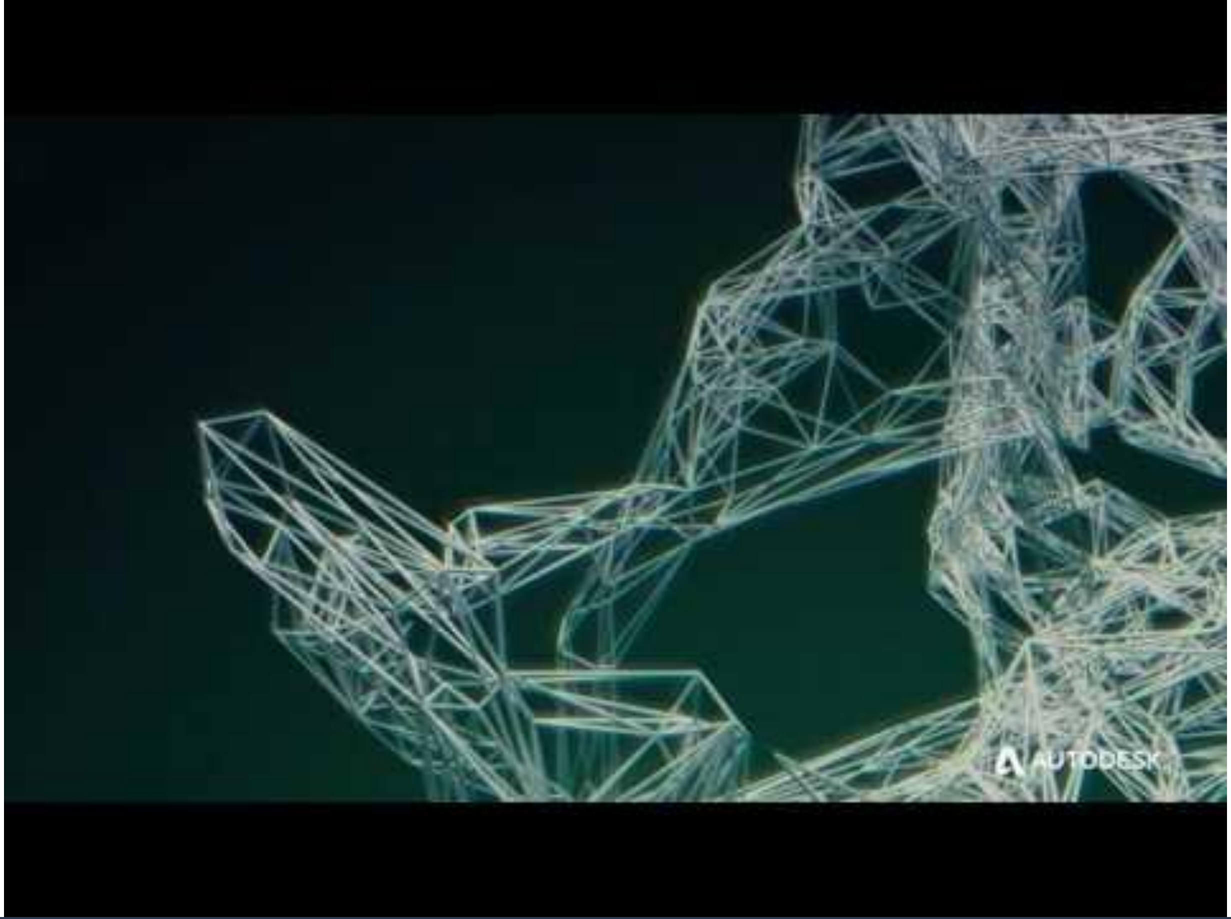
Parametric Design

Parametric Design (Architecture)

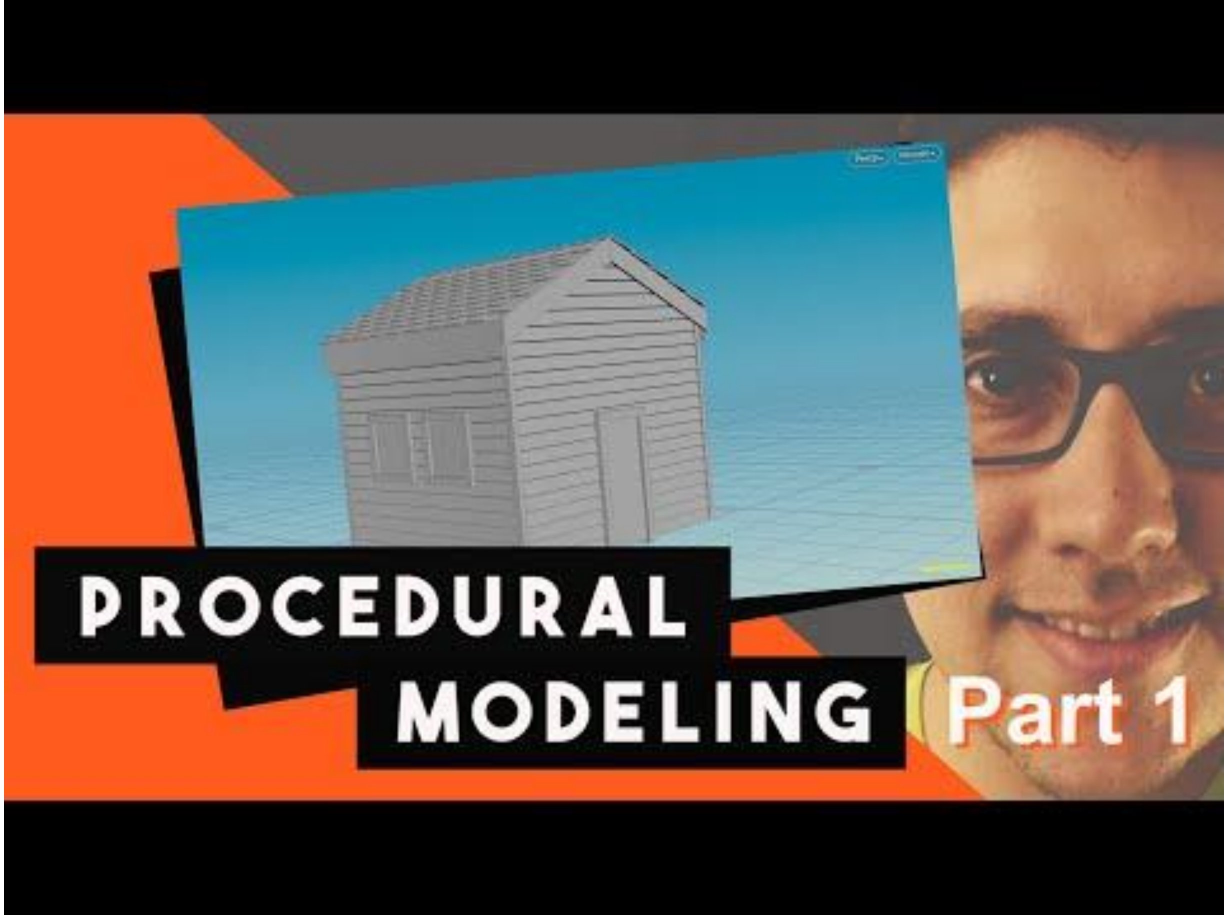




Generative Design (Architecture)



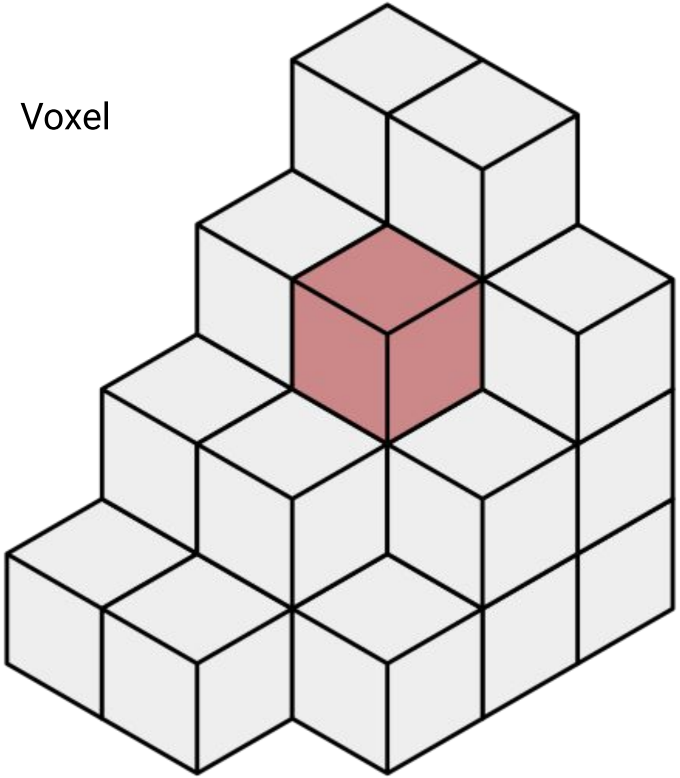
Generative Design

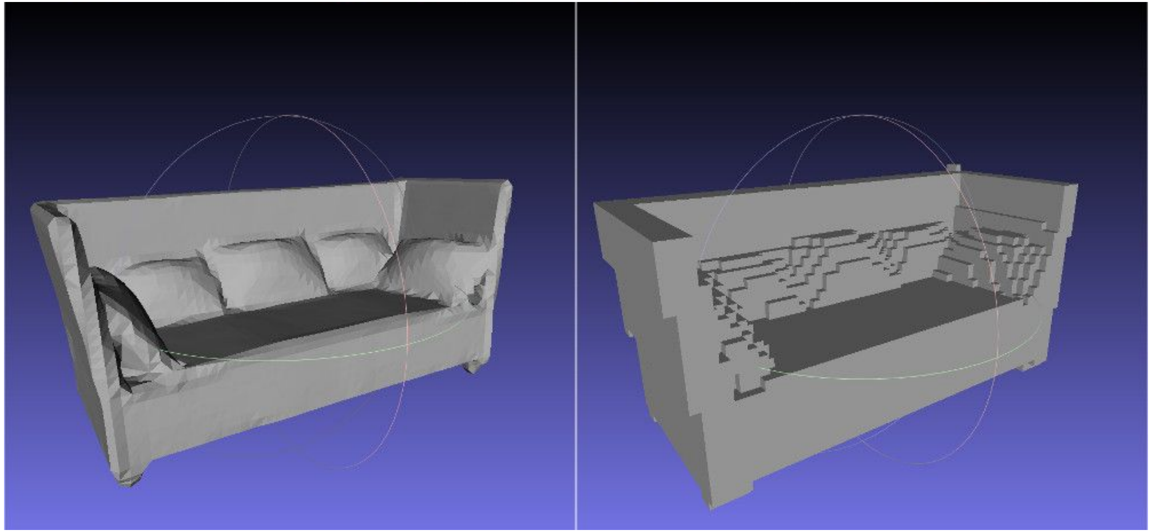


Procedural Design

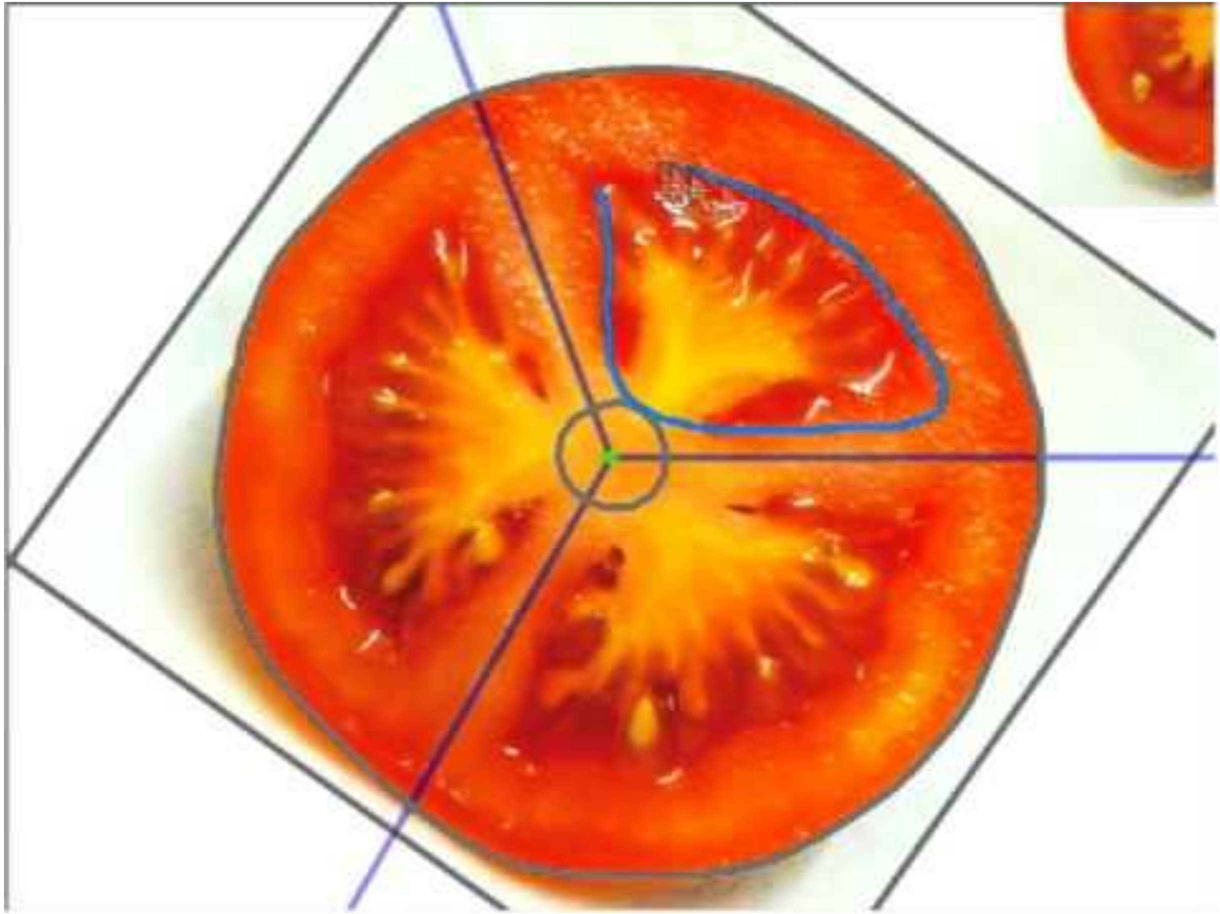
3D Model as Volume

Voxel



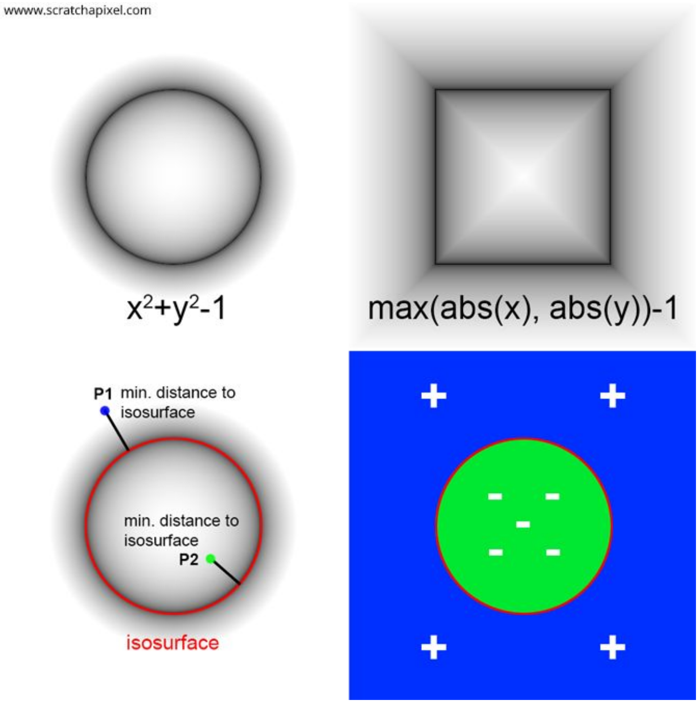


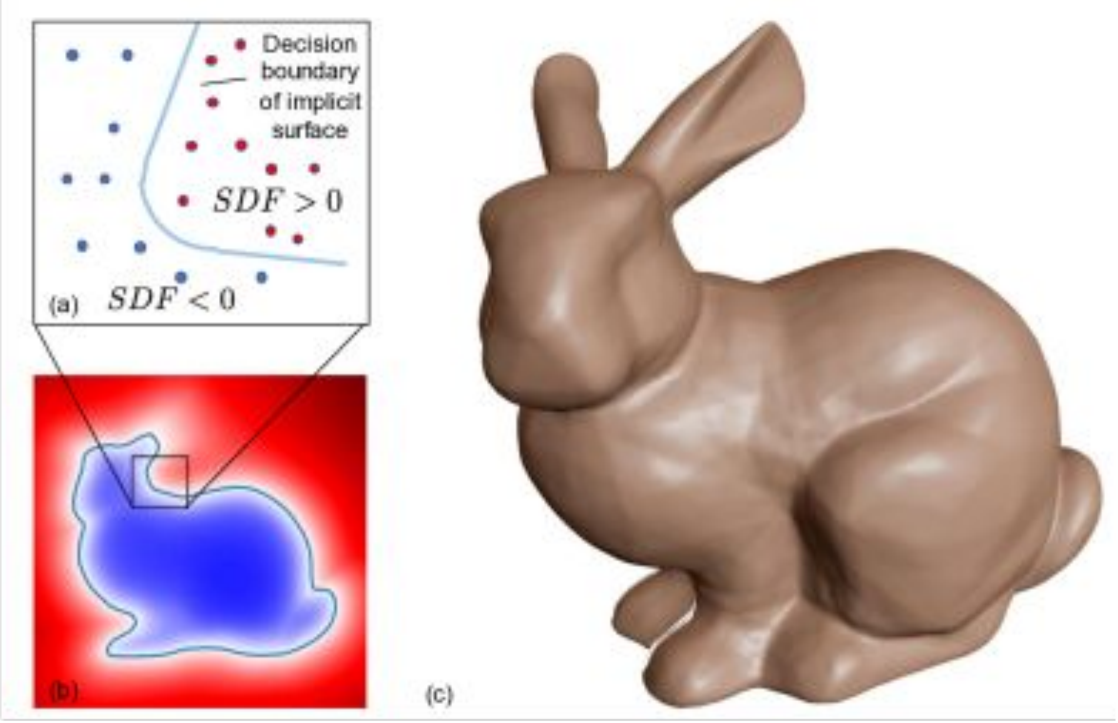
Mesh to Volume (Voxelization)

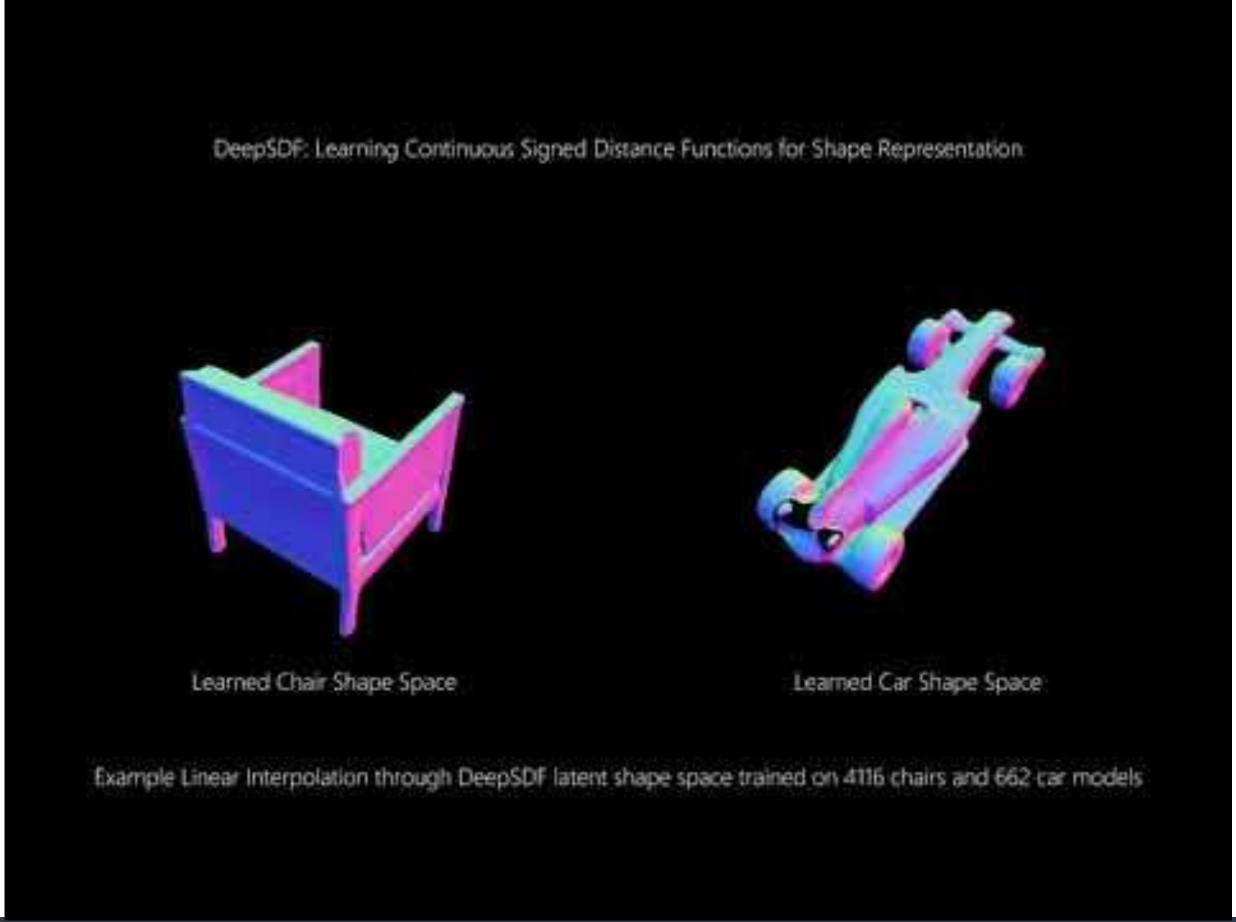


Volumetric Modeling

3D Volume: Model as Distance Function



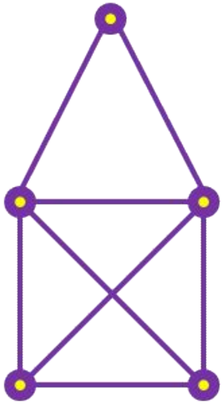




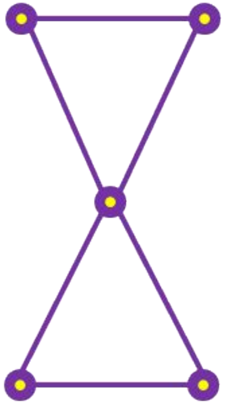
Learning SDFs

Deep Learning for 3D Models

3D Deep Learning Tasks
3D Mesh as Graph
Graphs Learning



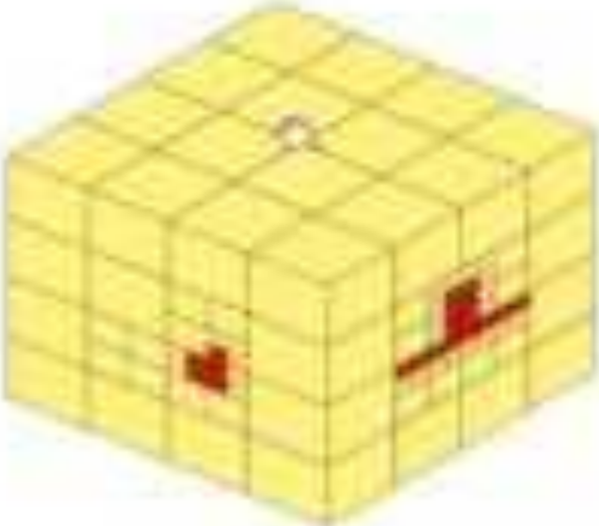
(1)



(1‘)

OctNet

: Pooling



Input

Fan et al. (2017)

Predicting 3D Volume from 2D Images

Edge Centric 3D Model

Face List

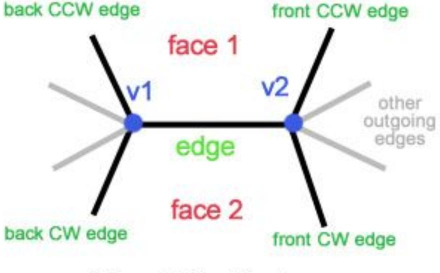
f0	4 8 9
f1	0 10 9
f2	5 10 11
f3	1 12 11
f4	6 12 13
f5	2 14 13
f6	7 14 15
f7	3 8 15
f8	4 16 19
f9	5 17 16
f10	6 18 17
f11	7 19 18
f12	0 23 20
f13	1 20 21
f14	2 21 22
f15	3 22 23

Edge List

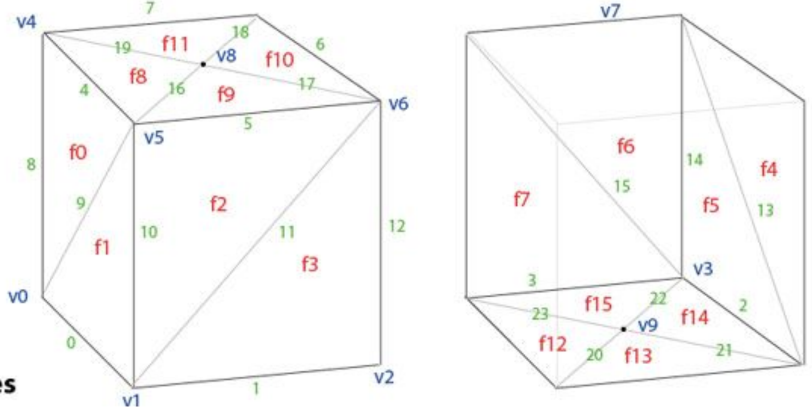
e0	v0 v1	f1 f12	9 23 10 20
e1	v1 v2	f3 f13	11 20 12 21
e2	v2 v3	f5 f14	13 21 14 22
e3	v3 v0	f7 f15	15 22 8 23
e4	v4 v5	f0 f8	19 8 16 9
e5	v5 v6	f2 f9	16 10 17 11
e6	v6 v7	f4 f10	17 12 18 13
e7	v7 v4	f6 f11	18 14 19 15
e8	v0 v4	f7 f0	3 9 7 4
e9	v0 v5	f0 f1	8 0 4 10
e10	v1 v5	f1 f2	0 11 9 5
e11	v1 v6	f2 f3	10 1 5 12
e12	v2 v6	f3 f4	1 13 11 6
e13	v2 v7	f4 f5	12 2 6 14
e14	v3 v7	f5 f6	2 15 13 7
e15	v3 v4	f6 f7	14 3 7 15
e16	v5 v8	f8 f9	4 5 19 17
e17	v6 v8	f9 f10	5 6 16 18
e18	v7 v8	f10 f11	6 7 17 19
e19	v4 v8	f11 f8	7 4 18 16
e20	v1 v9	f12 f13	0 1 23 21
e21	v2 v9	f13 f14	1 2 20 22
e22	v3 v9	f14 f15	2 3 21 23
e23	v0 v9	f15 f12	3 0 22 20

Vertex List

v0	0,0,0	8 9 0 23 3
v1	1,0,0	10 11 1 20 0
v2	1,1,0	12 13 2 21 1
v3	0,1,0	14 15 3 22 2
v4	0,0,1	8 15 7 19 4
v5	1,0,1	10 9 4 16 5
v6	1,1,1	12 11 5 17 6
v7	0,1,1	14 13 6 18 7
v8	.5,.5,0	16 17 18 19
v9	.5,.5,1	20 21 22 23



Winged Edge Structure



Winged-Edge Meshes

Graphs for the Deep Learning Era

Graph Neural N

Hidden state:

$$\mathbf{h}_v = f(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]})$$

$$\mathbf{H}^{t+1} = F(\mathbf{H}^t, \mathbf{X})$$

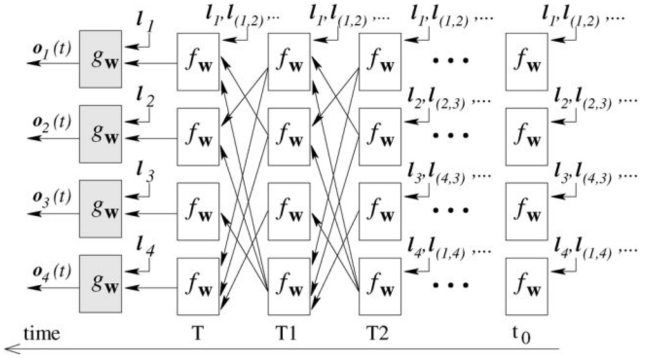
$$\mathbf{o}_v = g(\mathbf{h}_v, \mathbf{x}_v)$$

Output:

$$loss = \sum_{i=1}^{p} (\mathbf{t}_i - \mathbf{o}_i)$$

Node labeling

Loss:



[Scarselli ‘09]

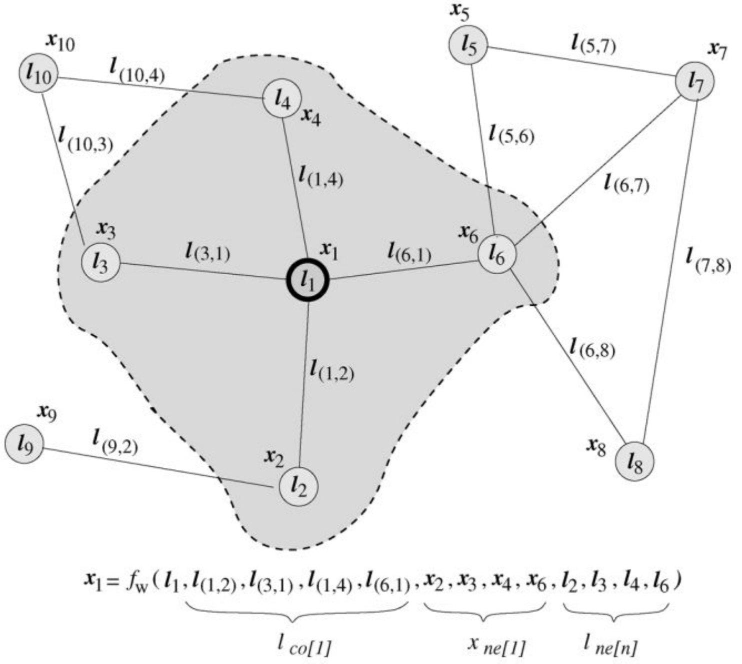
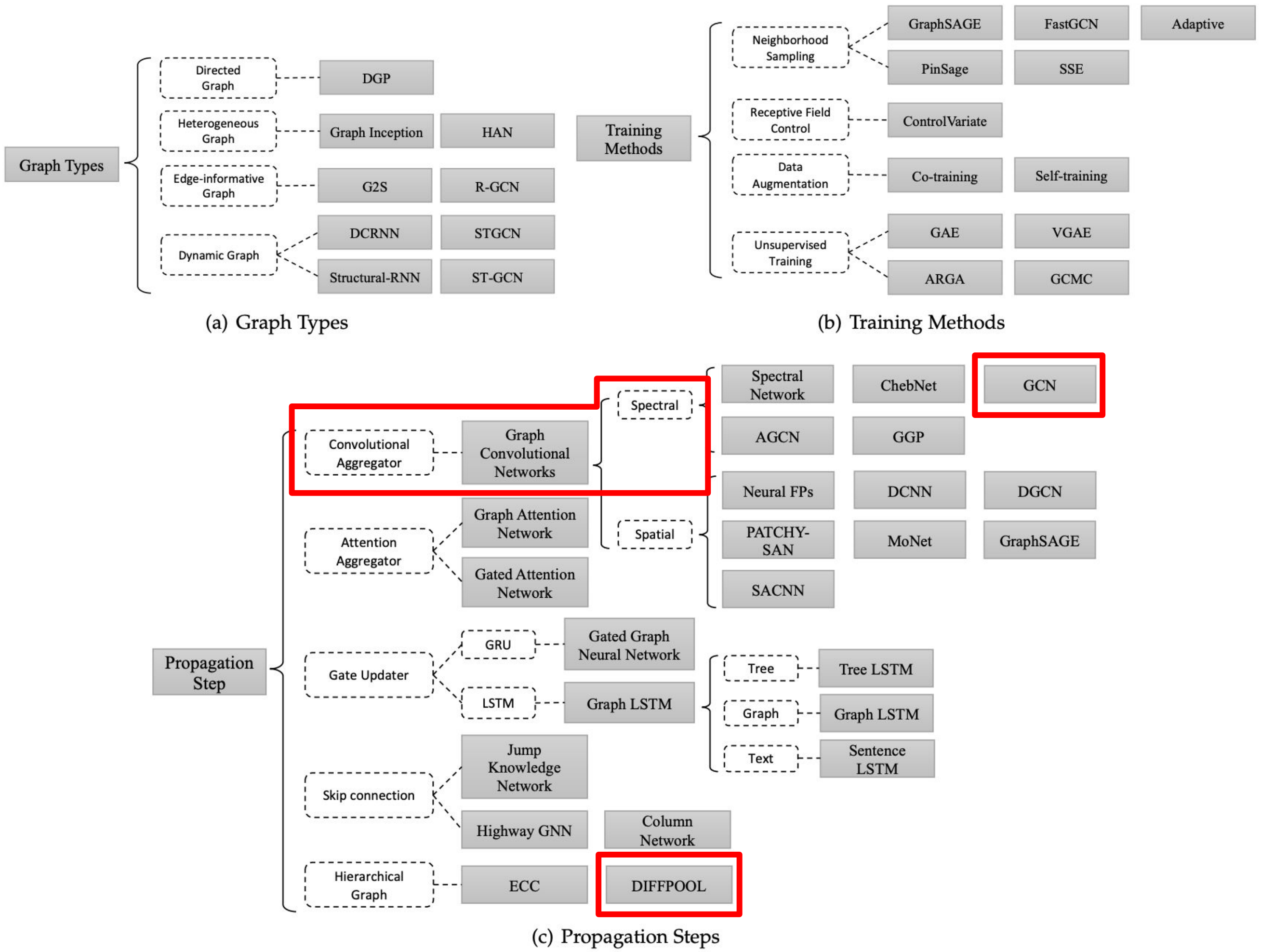


Fig. 2. Graph and the neighborhood of a node. The state x_1 of the node 1 depends on the information contained in its neighborhood.



Us, today.

Fig. 2. An overview of variants of graph neural networks.

[Zhou '18]

Graph Convolutional Networks

Popularized by: Kipf and Welling ‘16, ICLR ‘17
But it existed long before...

Graph Convolution Layer:

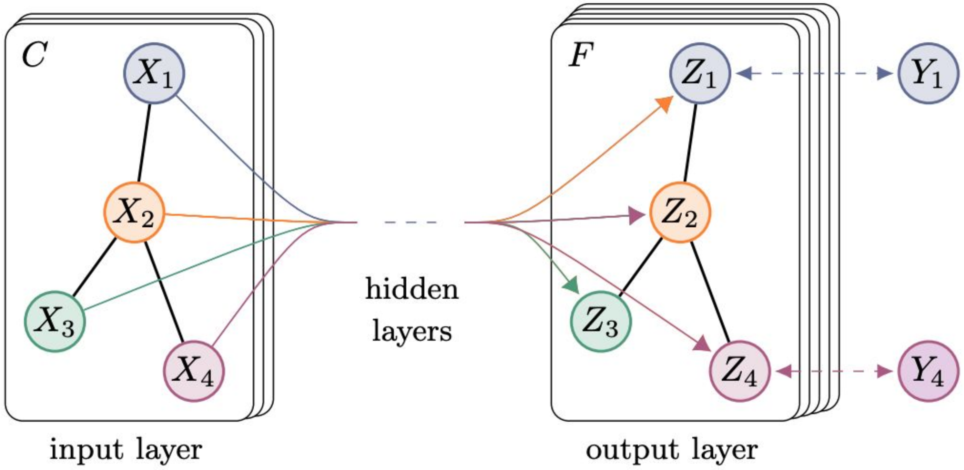
hidden activation adjacency weights

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

$$\tilde{A} = A + I_N \qquad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

Learn linear transformation of node features (embeddings), add non-linear activation.

Why “**Convolution**”?



(a) Graph Convolutional Network

Graph Convolutions

Symmetric normalized graph **Laplacian**:

$$\mathbf{L} = \mathbf{I_n} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

A (adjacency) 1s or 0s, its diagonal is all 0s
D degree matrix.

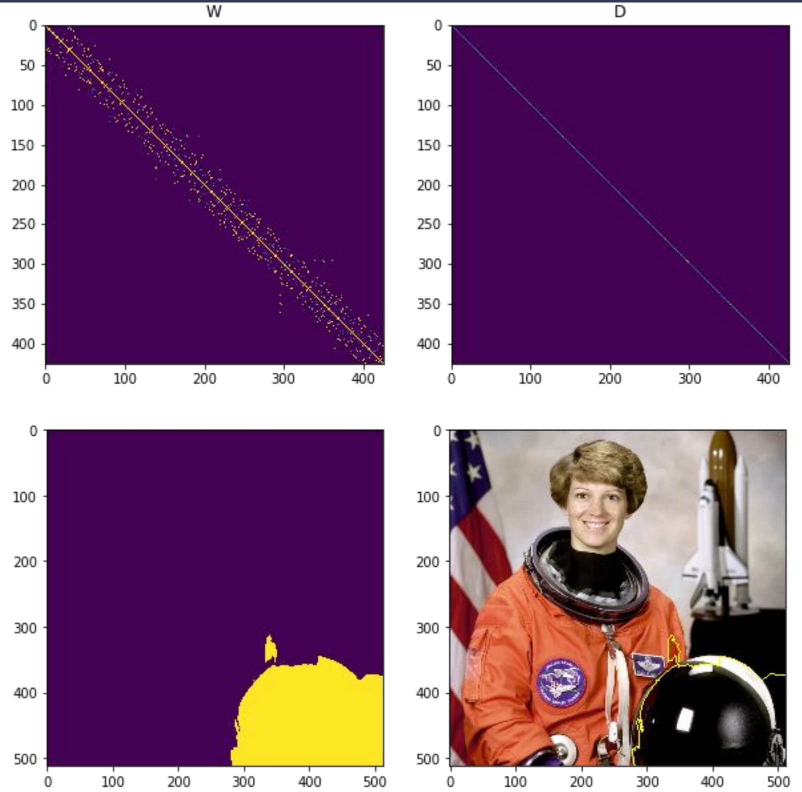
What is it useful for??

Well, it's actually useful for Segmentation!
[Shi, Jianbo, Malik '00] **Normalized Cuts**

Taking the second smallest eigenvector from:

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \quad \mathbf{\Lambda}_{ii} = \lambda_i$$

(In norm-cuts the non-sym Lap. is L = D - A)



Graph Convolutions

Graph Laplacian spectral analysis (e.g. eigen decomposition) exposes structure.

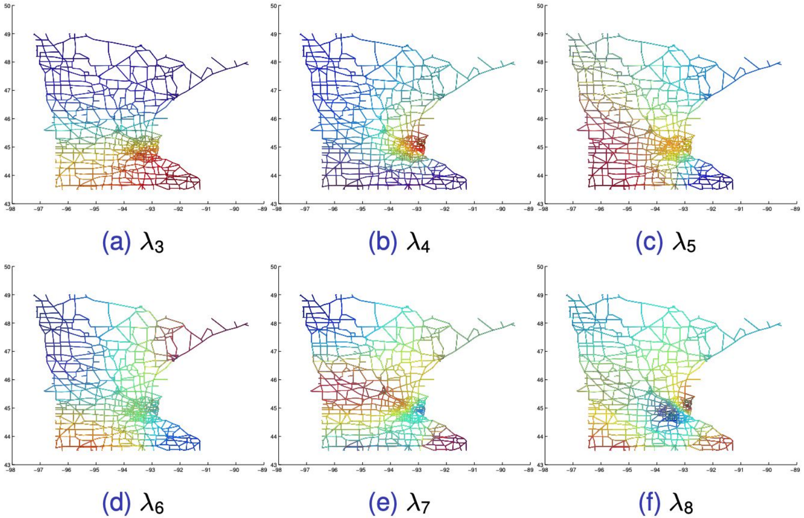


Figure : Eigenfunctions corresponding to the first six nonzero eigenvalues.
Minnesota road graph (2642 vertices)

Norbert Wiener Center for Harmonic Analysis and Applications

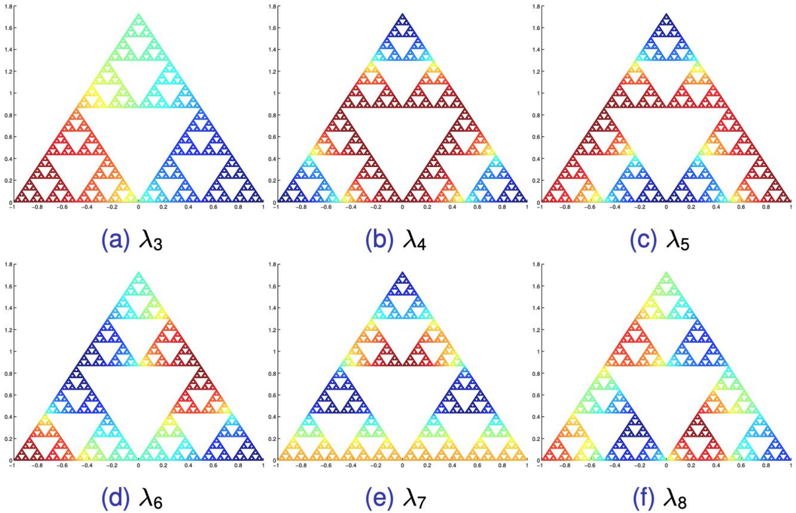


Figure : Eigenfunctions corresponding to the first six nonzero eigenvalues.
Level-8 graph approximation to Sierpinski gasket (9843 vertices)

Norbert Wiener Center for Harmonic Analysis and Applications

[Begué ‘14]

Graph Convolutions

Graph Fourier Transform: (spectral / eigen linear basis transform) [Wu '19]

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$$

$$\mathscr{F}(\mathbf{x}) = \mathbf{U}^T\mathbf{x} \qquad \mathscr{F}^{-1}(\hat{\mathbf{x}}) = \mathbf{U}\hat{\mathbf{x}}$$

$$\mathbf{x} = \sum_i \hat{\mathbf{x}}_i \mathbf{u}_i$$

Graph filter, via convolution theorem:

$$\mathbf{g} \in \mathbf{R}^N$$

$$\mathbf{x} *_G \mathbf{g} = \mathscr{F}^{-1}(\mathscr{F}(\mathbf{x}) \odot \mathscr{F}(\mathbf{g}))$$
$$= \mathbf{U}(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g})$$

Linear simplification: $\mathbf{g}_\theta = diag(\mathbf{U}^T\mathbf{g}) \longrightarrow \mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x}$

Graph Convolutions

[Kipf '16]

Generalized convolution: $$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x}$$

Solving for g_θ requires K-th order polynomials in the Laplacian, and depends on nodes that are K steps away from the central node (K-th-order neighborhood).

[Kipf & Welling '16] suggest taking K = 1, and end up with the simpler linear transformation:

$$g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta \qquad X \in \mathbb{R}^{N \times C}$$

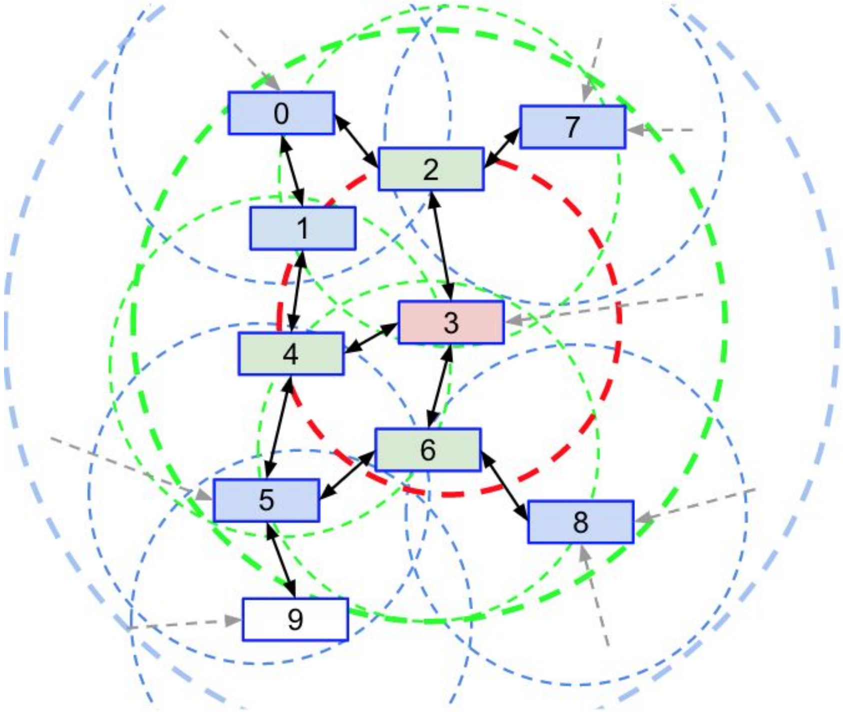
It is **differentiable** in Θ, and **fast to compute** both FF and Backprop.

And the beauty is that it follows CNN paradigm: **Stack layers**!

Receptive Field Propagation

By stacking GC layers we increase the receptive field of higher-level neurons.

However there's one big problem ---



Receptive Field Propagation

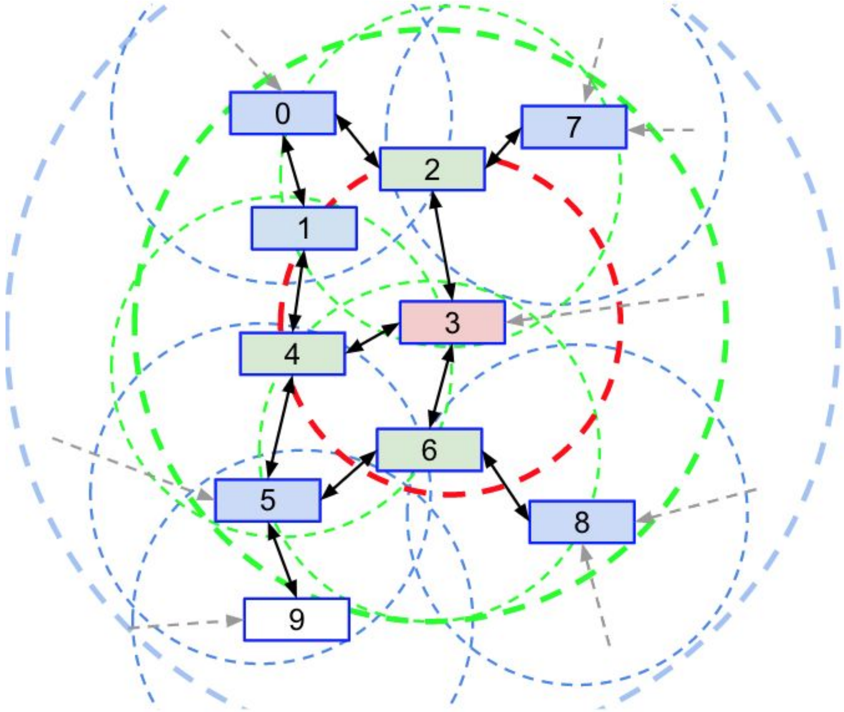
By stacking GC layers we increase the receptive field of higher-level neurons.

However there's one big problem ---

Graph *layout* stays the same layer-to-layer.
We have the *same amount of features* to learn in each layer === it doesn't scale.

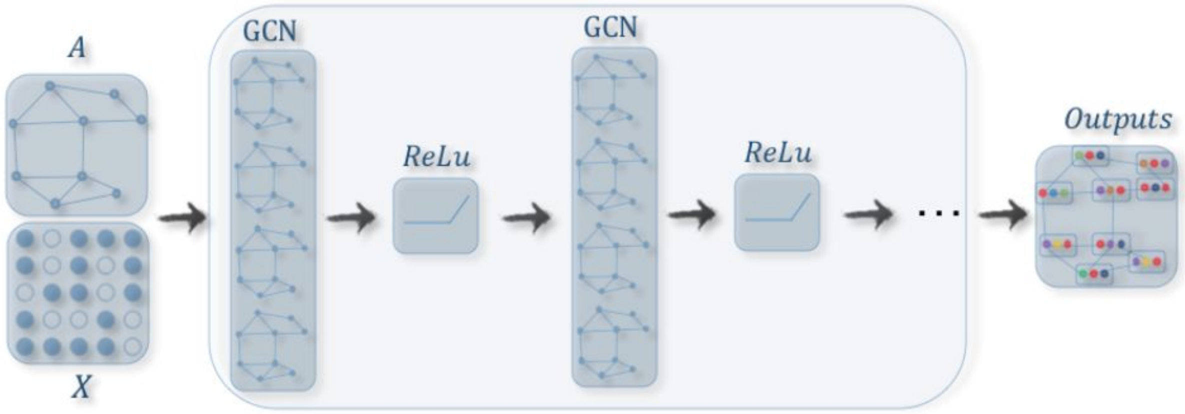
It's essentially a MLP.

(Quick to overfit, bloated, data hungry models)

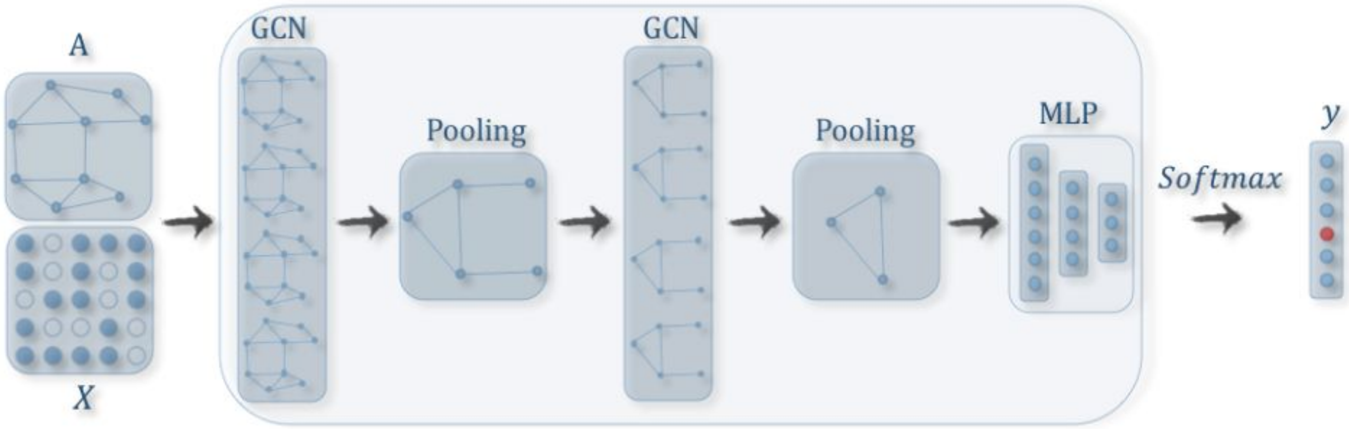


[Wu '19]

Node
classification



Graph
classification



DiffPool and conv-pool

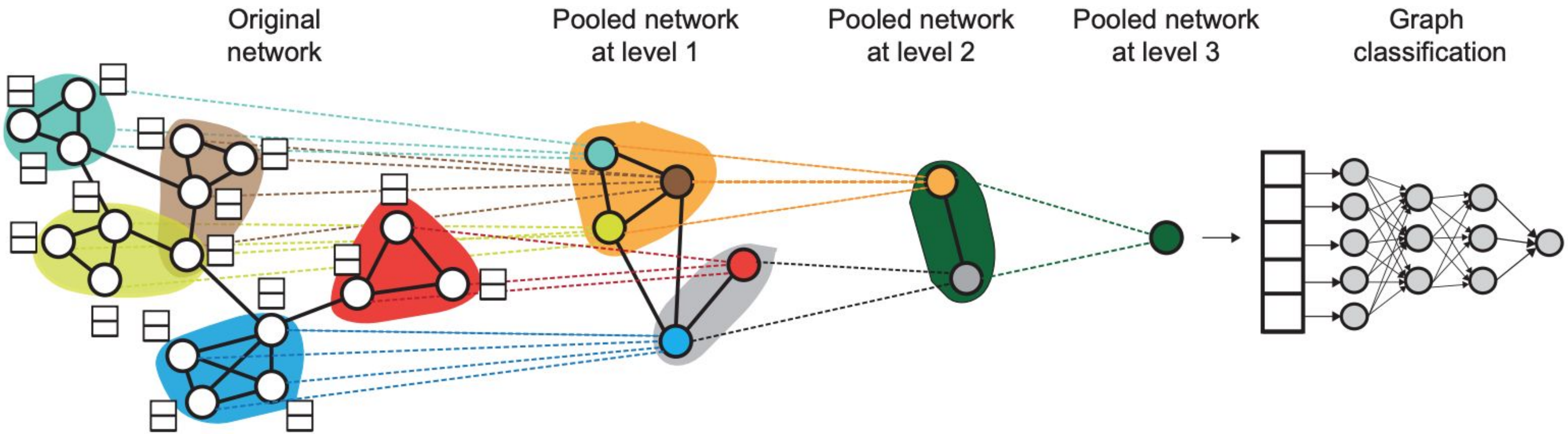


Figure 1: High-level illustration of our proposed method DIFFPOOL. At each hierarchical layer, we run a GNN model to obtain embeddings of nodes. We then use these learned embeddings to cluster nodes together and run another GNN layer on this coarsened graph. This whole process is repeated for L layers and we use the final output representation to classify the graph.

[Ying '18]

Learning How to "Pool" a Graph

[Ying '18]

Select the nodes (via indicator matrix) that will go to the next cluster.

Learn this as a *parametric assignment*.
E.g. the assignment is dependant on the node features + neighbor features.

$$A^{(i+1)} = S^{(i)\top} A^{(i)} S^{(i)}$$

$$S^{(i)} = \text{softmax}\left(\text{GNN}_{\text{pool}}\left(A^{(i)}, F^{(i)}\right)\right)$$

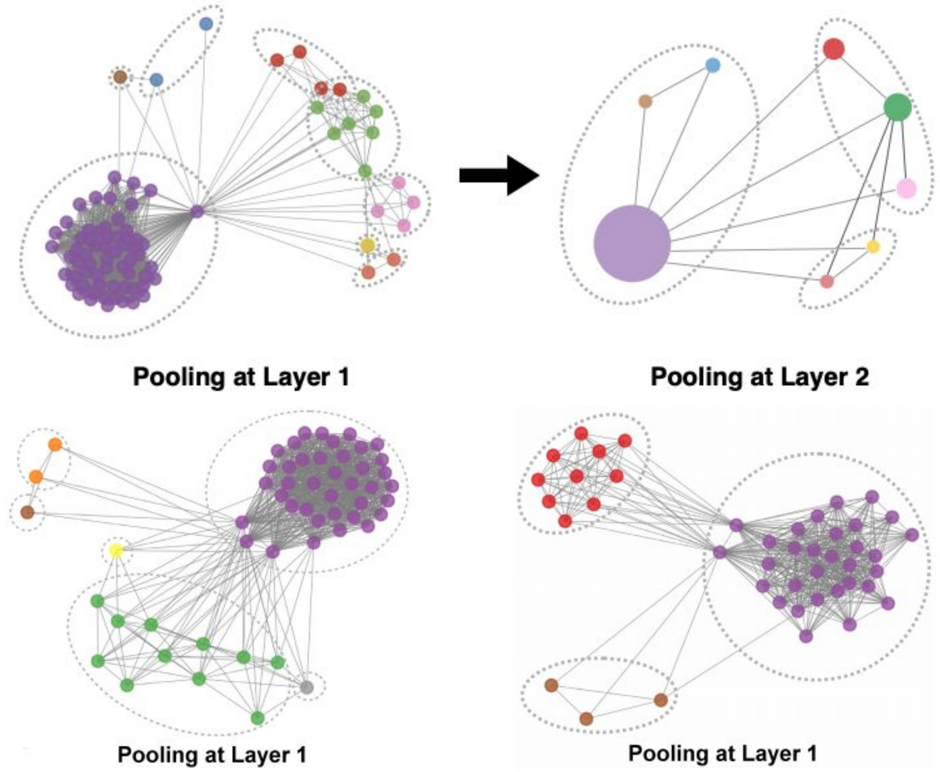
[Kipf et al]

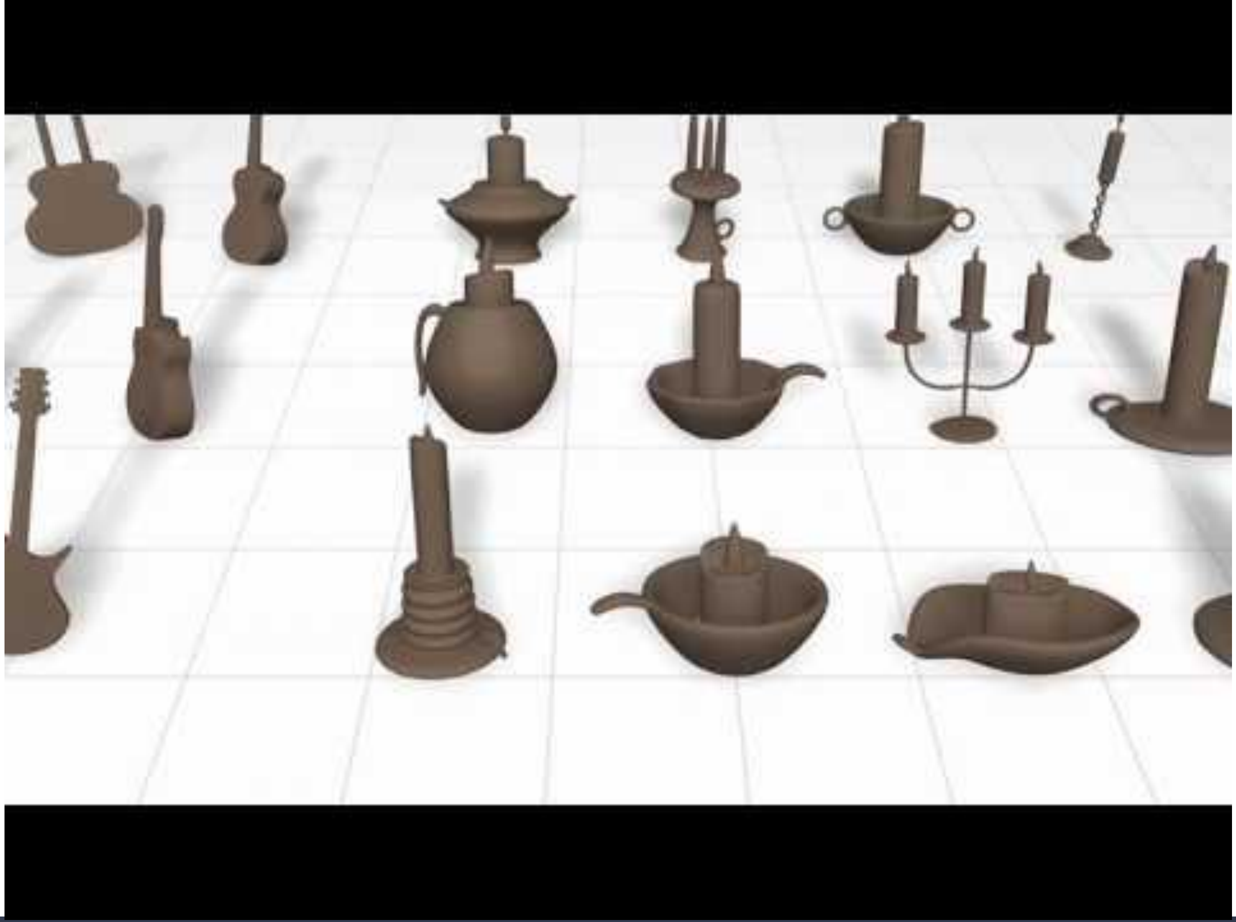
Encourage neighbors to *stick together*:

$$L_{\text{LP}} = ||A^{(l)}, S^{(l)} S^{(l)^T}||_F$$

Enforce *singular assignment* via entropy:

$$L_{\text{E}} = \frac{1}{n} \sum_{i=1}^{n} H(S_i)$$

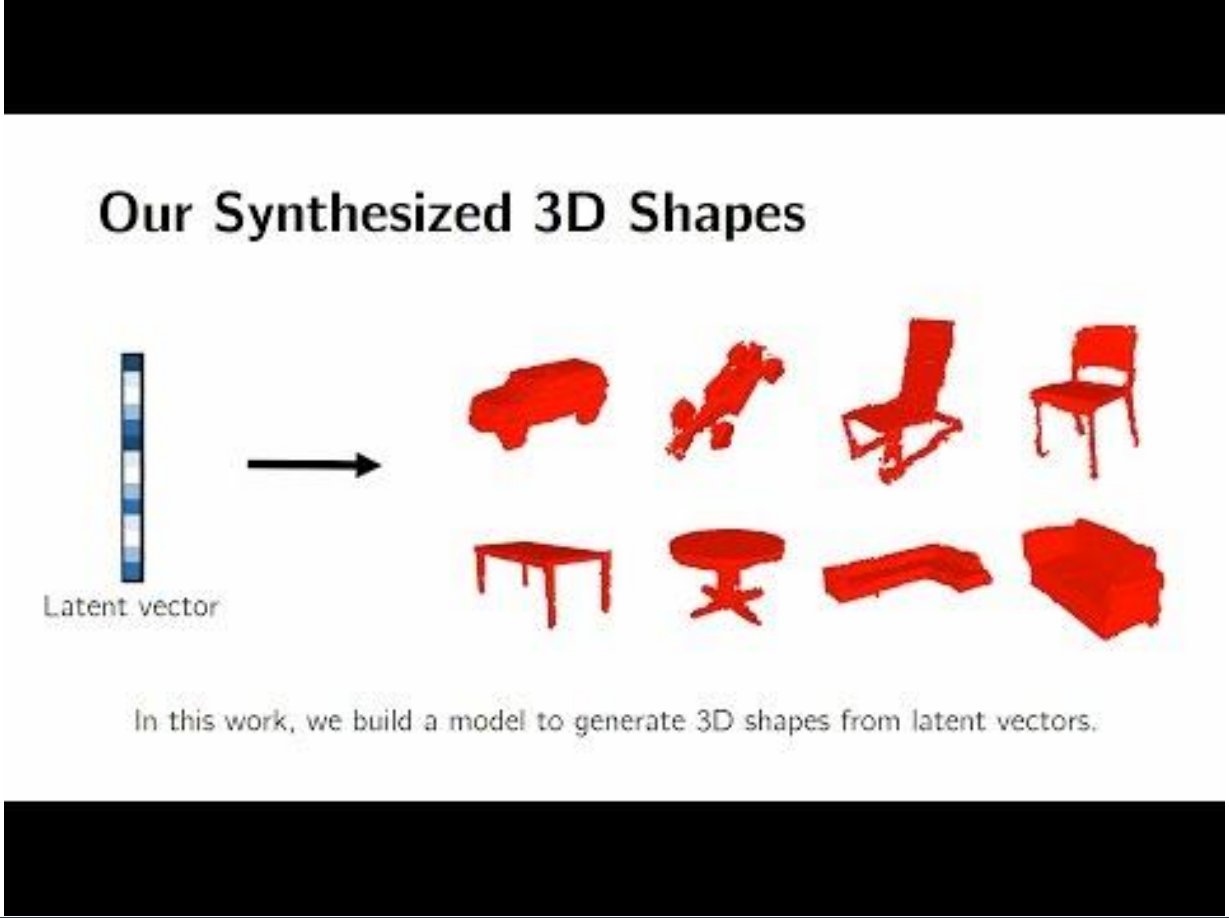




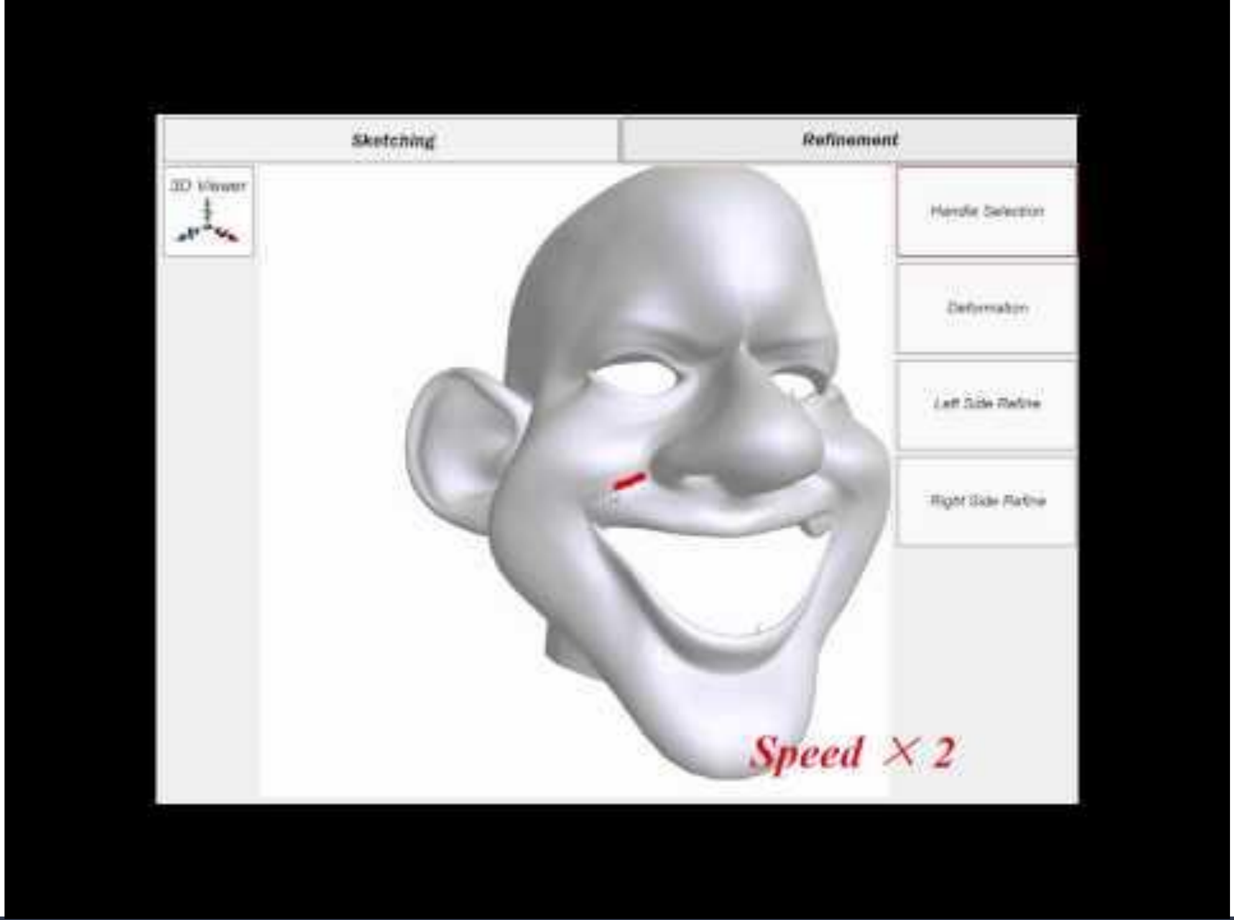
Graph Convolutions for 3D Mesh Segmentation

Generative 3D Models

3DGAN
Caricature to 3D
Image to 3D
HoloGAN

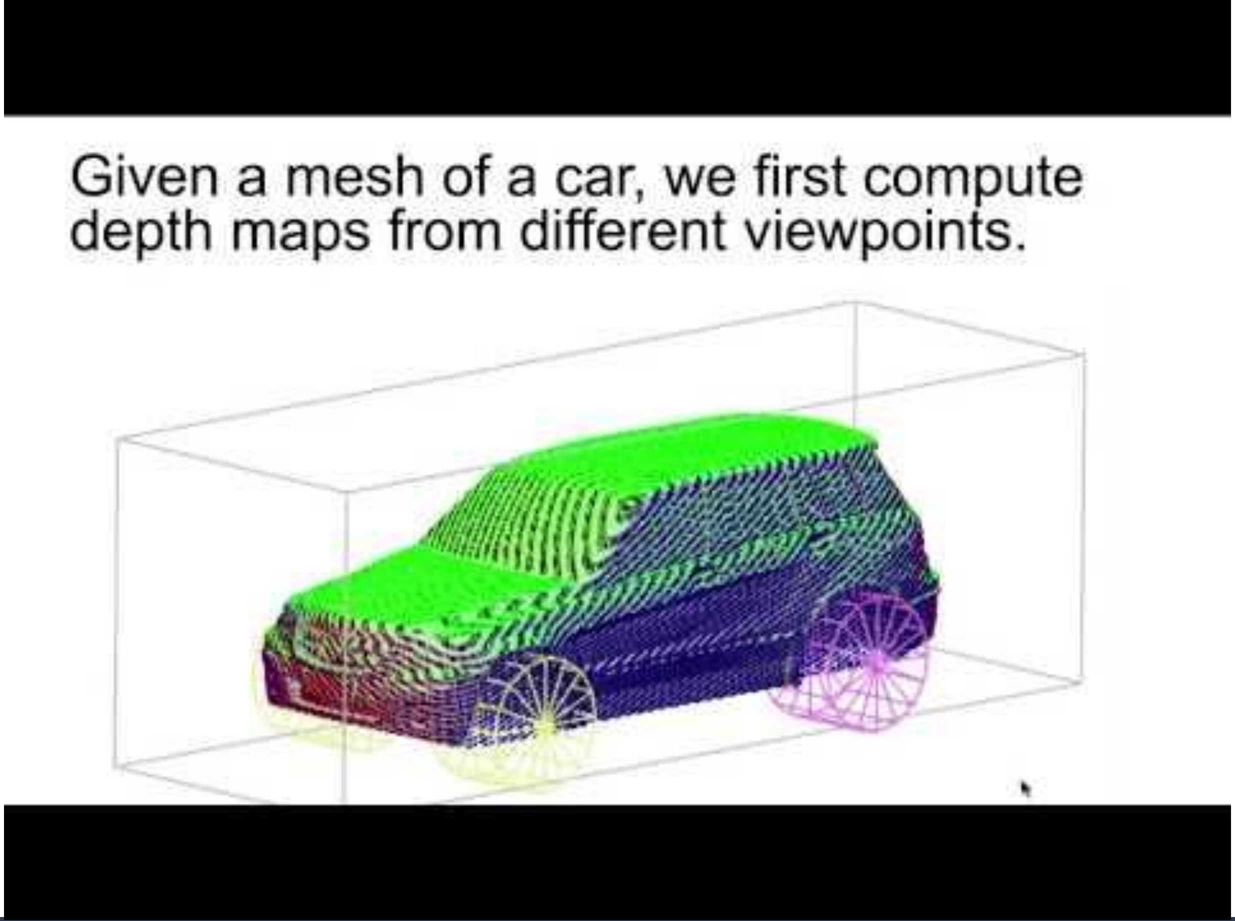


Latent Vector 3D Reconstruction: 3DGAN

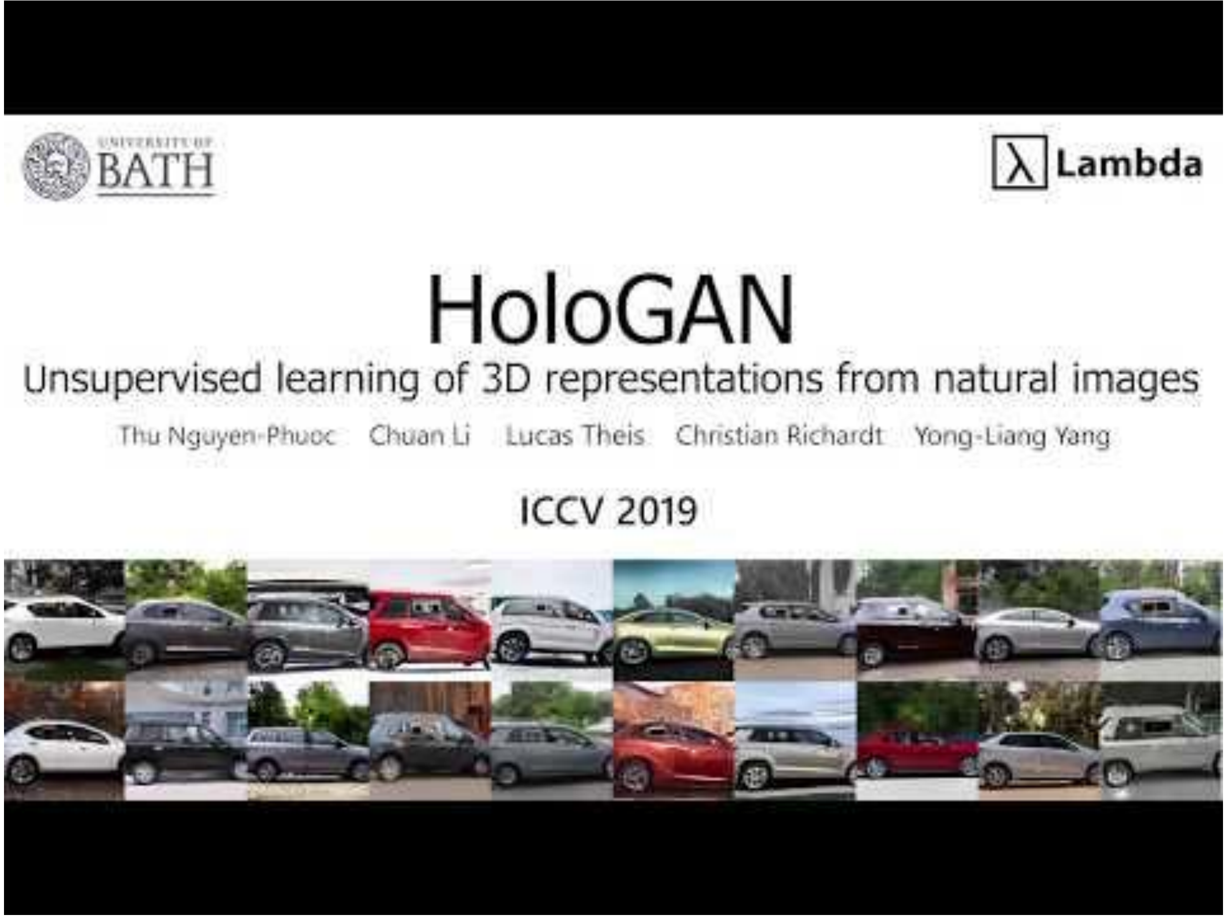

Sketching
Refinement
3D Viewer
Handle Selection
Deformation
Left Side Refine
Right Side Refine
Speed × 2

Sketch-to-3D

3D from Image



Latent 3D Representation with Projection Autoencoders



3D convolutions

End of Slides