Analytic Marching: An Analytic Meshing Solution from Deep Implicit Surface Networks

Jiabao Lei and Kui Jia

To Appear in International Conference on Machine Learning (ICML) 2020

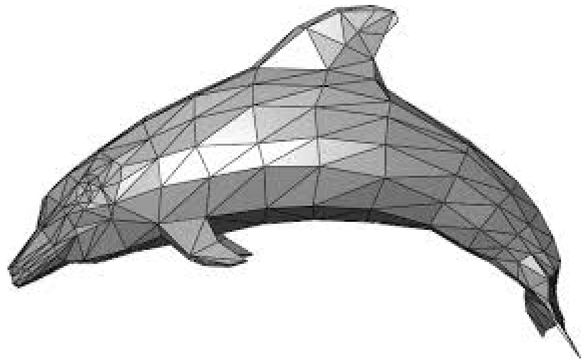
Introduction - Surface representation

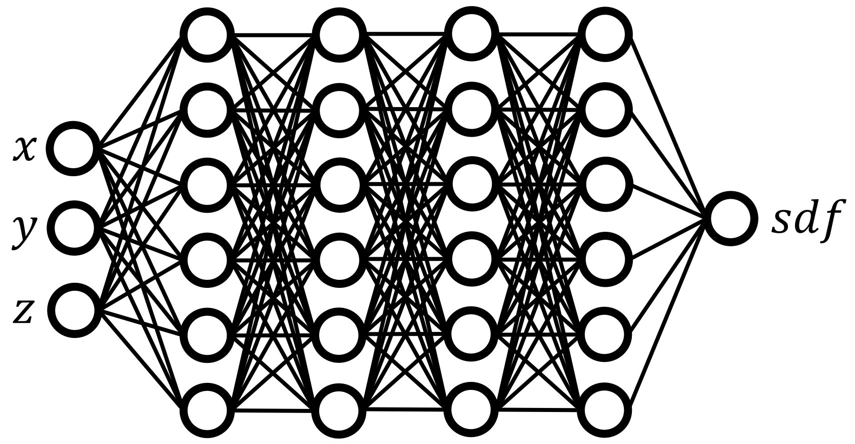
Explicit representation

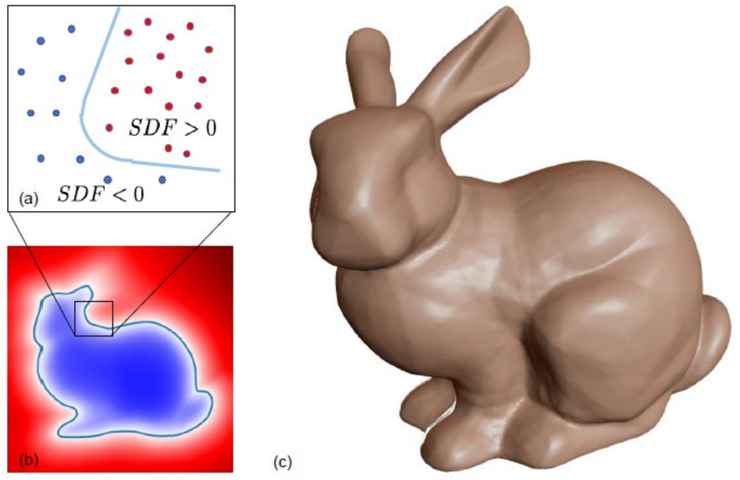
- ✓ Polygonal mesh is a piecewise linear approximation to the 3D surface.

Implicit representation

- ✓ Signed Distance Function (SDF) , Occupancy field, etc
- ✓ The surface is represented as the zero-level isosurface of an SDF.
- ✓ Deep SDFs are usually implemented as Multi-Layer Perceptrons (MLPs) with Rectified Linear Units (ReLUs).







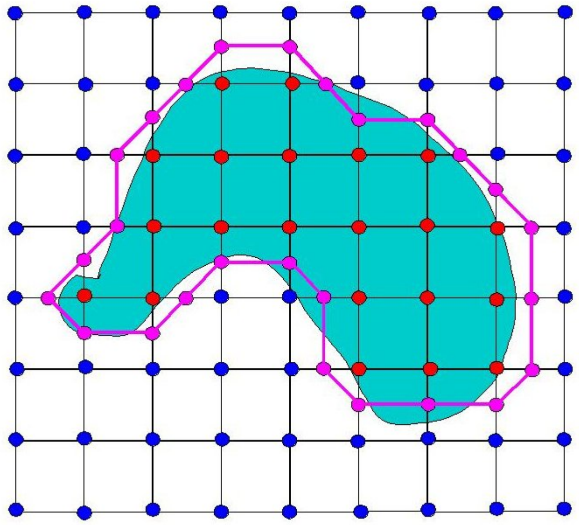
Introduction - Isosurface extraction

Isosurface extraction (meshing, *e.g., via marching cubes*)

- ✓ **Definition:** The conversion from an implicit field to an explicit surface mesh.
- ✓ Existing methods of deep implicit mesh recovery typically take a final step of Marching Cubes (MC).

$$\begin{tikzcd}\boxed{\textbf{Input}} \arrow[r] & \boxed{\textbf{Implicit function}} \arrow[r, "\textbf{Isosurface extraction}"] & \boxed{\textbf{Mesh}}\end{tikzcd}$$

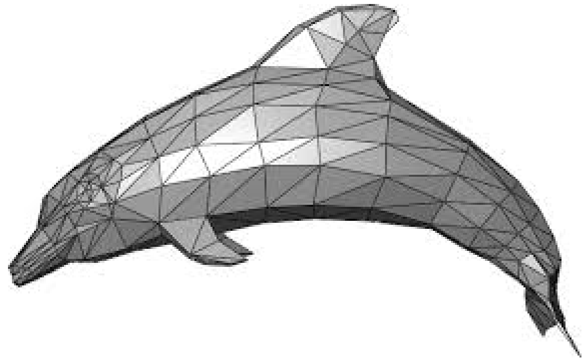
- ✓ While promising, it suffers from loss of precision learned in the MLPs, due to the discretization nature of Marching Cubes.
- ✓ They require lots of points sampled from 3D space, which takes time, and the resulting mesh is just an approximation to the underlying isosurface.

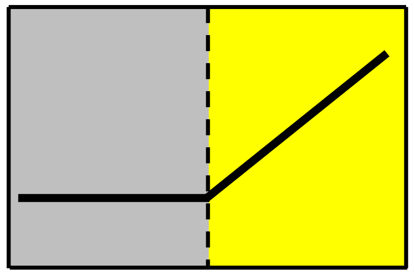

2D MC illustration

Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*



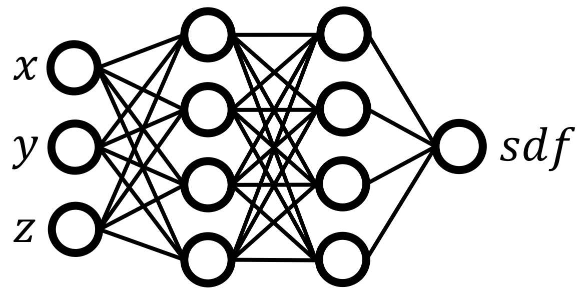


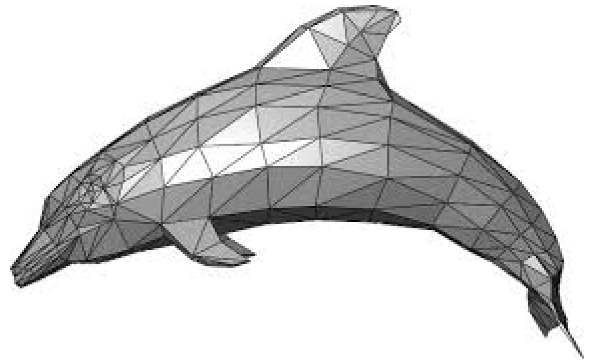
ReLU

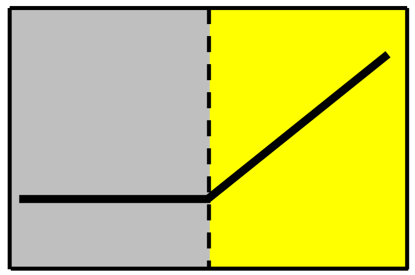
Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*



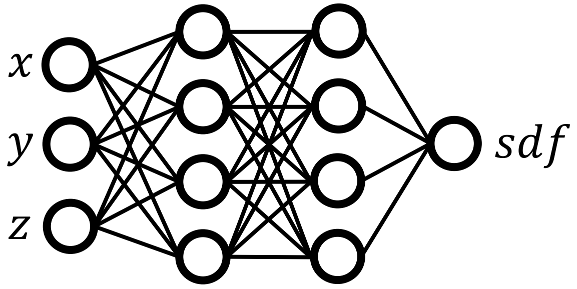


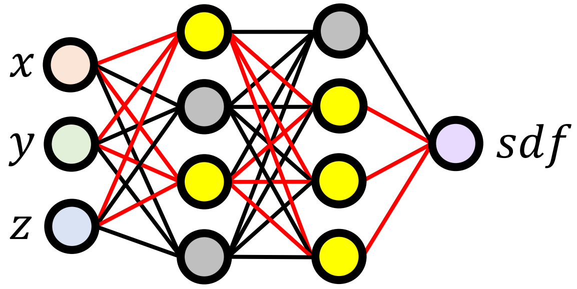

ReLU

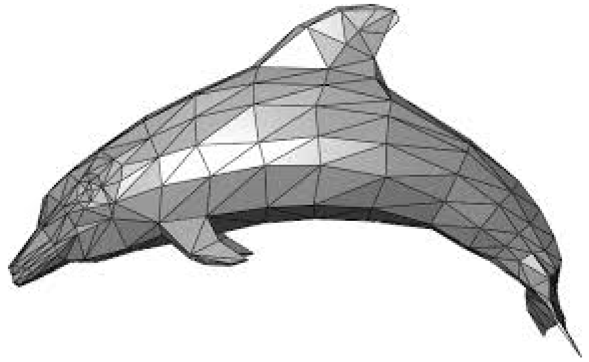
Motivation – connecting the properties of local linearity in two worlds

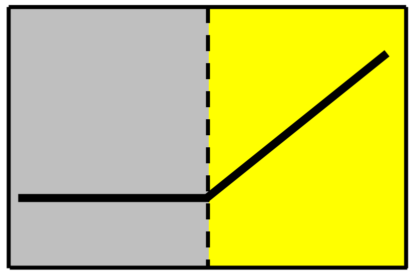
Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*





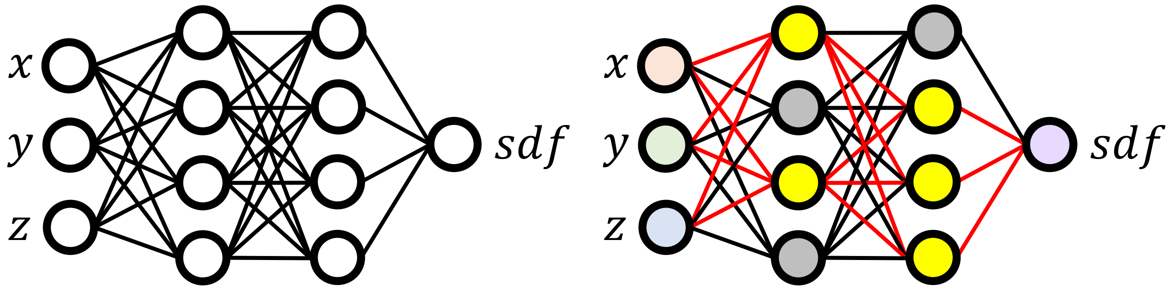



ReLU

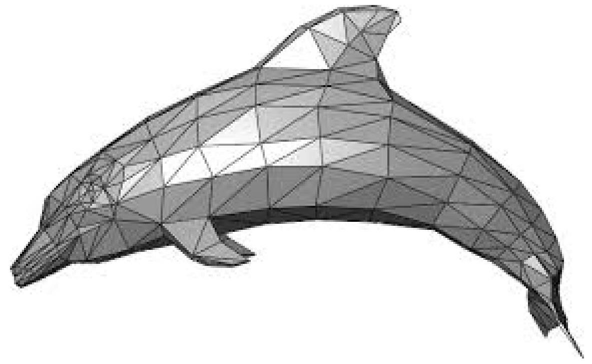
Motivation – connecting the properties of local linearity in two worlds

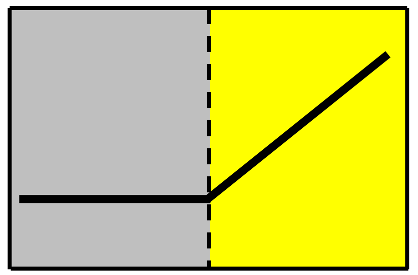
Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*



state vector = $[\underbrace{1\ 0\ 1\ 0}\ \underbrace{0\ 1\ 1\ 1}]$

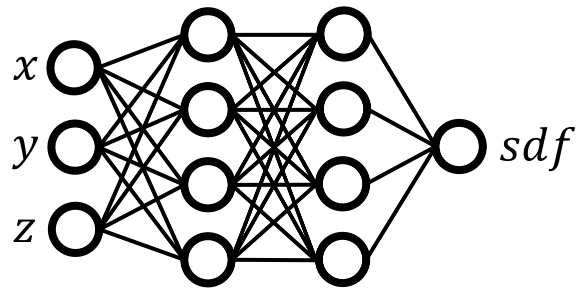


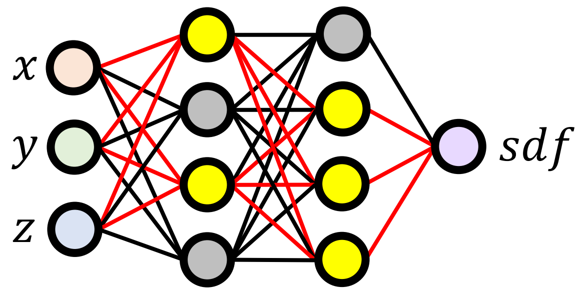

ReLU

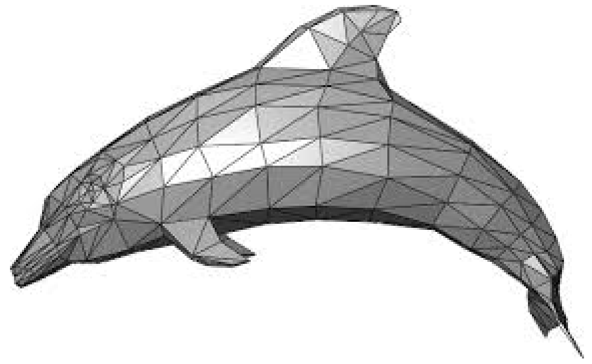
Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

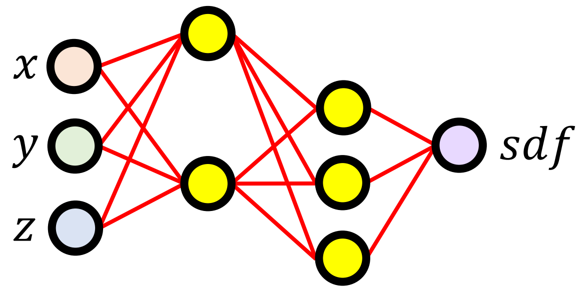
- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*

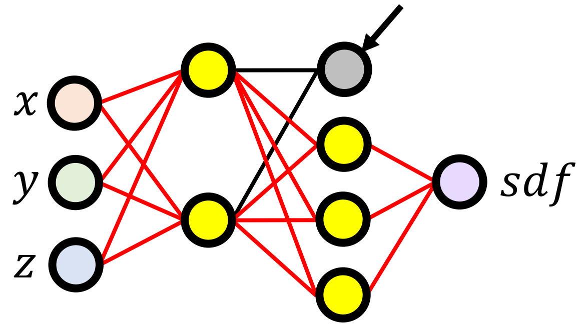


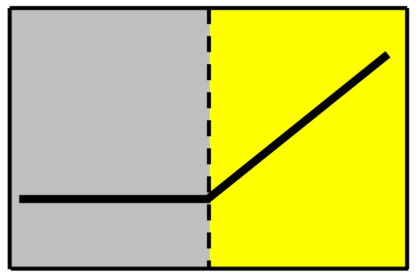




state vector = $[\underbrace{1\ 0\ 1\ 0}\ \underbrace{0\ 1\ 1\ 1}]$





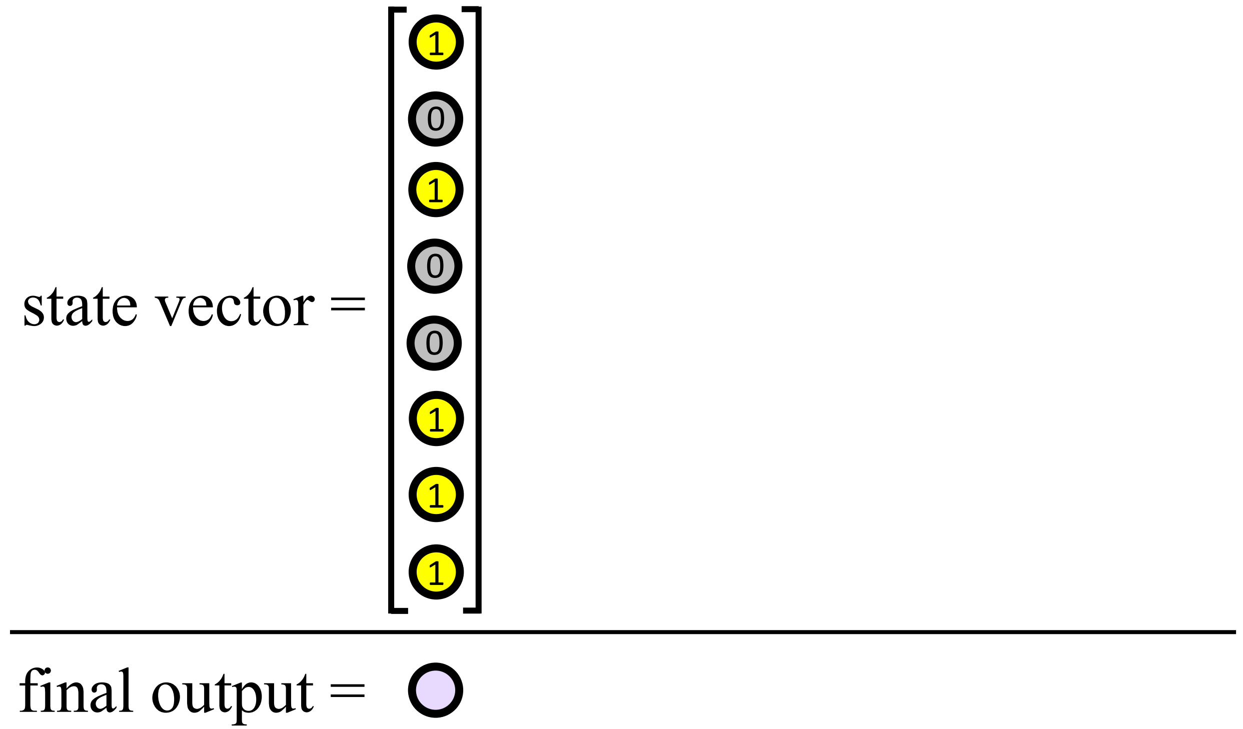


ReLU

Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*



Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*

$$
\text{state vector} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}
\begin{array}{l}
\rightarrow \boxed{a_{11}x + b_{11}y + c_{11}z + d_{11}} \\
\rightarrow \boxed{a_{12}x + b_{12}y + c_{12}z + d_{12}} \\
\rightarrow \boxed{a_{13}x + b_{13}y + c_{13}z + d_{13}} \\
\rightarrow \boxed{a_{14}x + b_{14}y + c_{14}z + d_{14}} \\
\rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} \\
\rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} \\
\rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} \\
\rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}}
\end{array}
$$

$$
\text{final output} = \bigcirc \rightarrow \boxed{a_F x + b_F y + c_F z + d_F}
$$

Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*

$$\text{state vector} = \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{array}\right] \begin{array}{l} \rightarrow \boxed{a_{11}x + b_{11}y + c_{11}z + d_{11}} > 0 \\ \rightarrow \boxed{a_{12}x + b_{12}y + c_{12}z + d_{12}} < 0 \\ \rightarrow \boxed{a_{13}x + b_{13}y + c_{13}z + d_{13}} > 0 \\ \rightarrow \boxed{a_{14}x + b_{14}y + c_{14}z + d_{14}} < 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} < 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \end{array}$$

$$\text{final output} = \bigcirc \rightarrow \boxed{a_F x + b_F y + c_F z + d_F} = 0$$

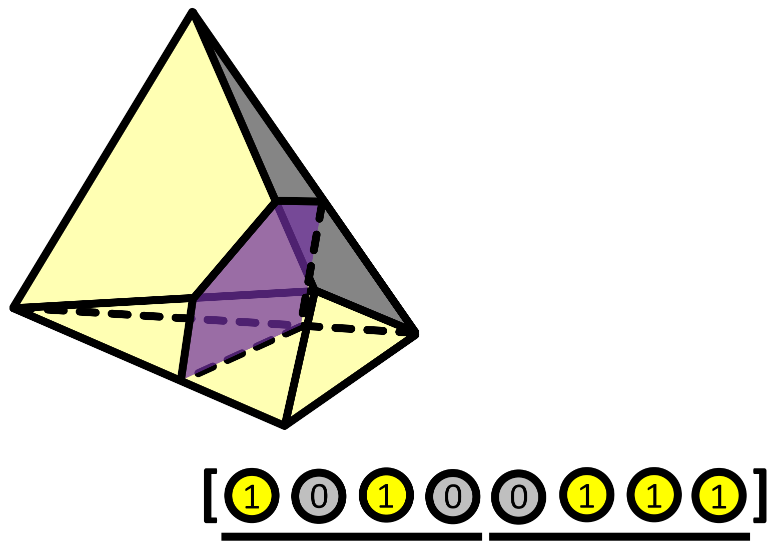
Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*

$$\text{state vector} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{array}{l} \rightarrow \boxed{a_{11}x + b_{11}y + c_{11}z + d_{11}} > 0 \\ \rightarrow \boxed{a_{12}x + b_{12}y + c_{12}z + d_{12}} < 0 \\ \rightarrow \boxed{a_{13}x + b_{13}y + c_{13}z + d_{13}} > 0 \\ \rightarrow \boxed{a_{14}x + b_{14}y + c_{14}z + d_{14}} < 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} < 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \end{array}$$

$$\text{final output} = \bigcirc \rightarrow \boxed{a_F x + b_F y + c_F z + d_F} = 0$$



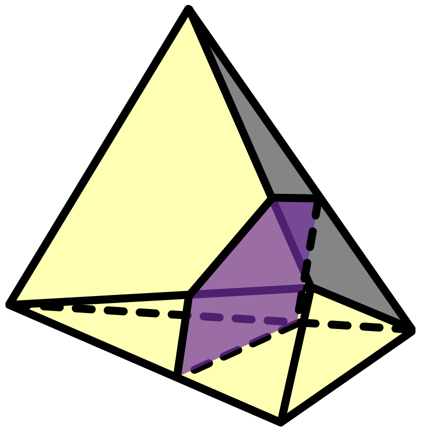
Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*

$$
\text{state vector} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}
\begin{array}{l}
\rightarrow \boxed{a_{11}x + b_{11}y + c_{11}z + d_{11}} > 0 \\
\rightarrow \boxed{a_{12}x + b_{12}y + c_{12}z + d_{12}} < 0 \\
\rightarrow \boxed{a_{13}x + b_{13}y + c_{13}z + d_{13}} > 0 \\
\rightarrow \boxed{a_{14}x + b_{14}y + c_{14}z + d_{14}} < 0 \\
\rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} < 0 \\
\rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \\
\rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \\
\rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0
\end{array}
$$

$$
\text{final output} = \bigcirc \rightarrow \boxed{a_F x + b_F y + c_F z + d_F} = 0
$$



$$[\underline{1\;0\;1\;0}\;\underline{0\;1\;1\;1}]$$

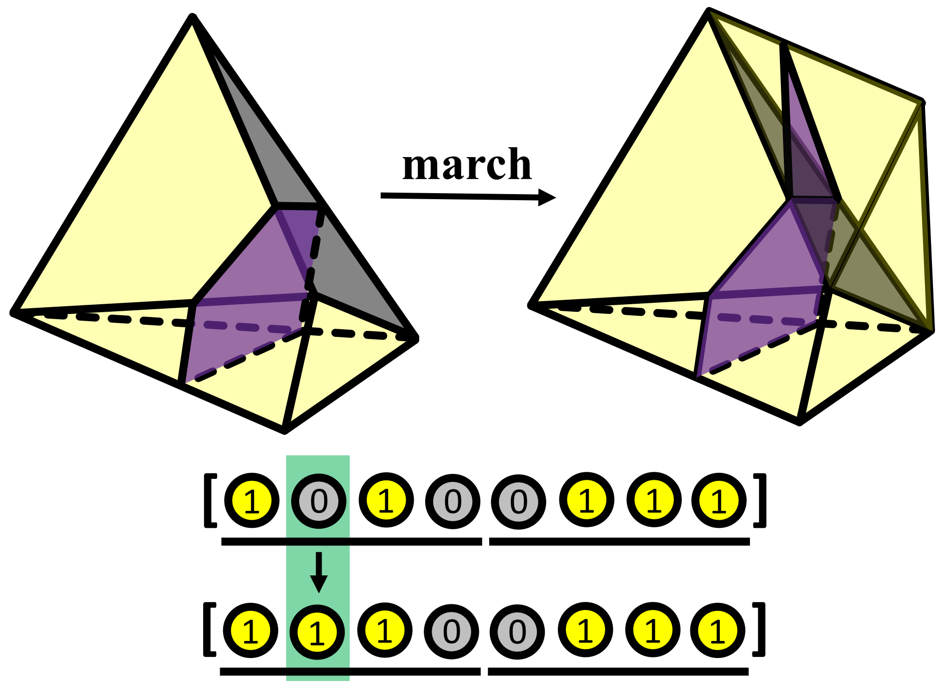
Motivation – connecting the properties of local linearity in two worlds

Piecewise linear property

- ✓ A polygonal mesh is a piece-wise linear approximation of object surface
- ✓ ReLU activation function is also piece-wise linear
- ✓ *Any relations between the properties of local linearity in the two worlds?*

$$\text{state vector} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{array}{l} \rightarrow \boxed{a_{11}x + b_{11}y + c_{11}z + d_{11}} > 0 \\ \rightarrow \boxed{a_{12}x + b_{12}y + c_{12}z + d_{12}} < 0 \\ \rightarrow \boxed{a_{13}x + b_{13}y + c_{13}z + d_{13}} > 0 \\ \rightarrow \boxed{a_{14}x + b_{14}y + c_{14}z + d_{14}} < 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} < 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \\ \rightarrow \boxed{a_{21}x + b_{22}y + c_{23}z + d_{24}} > 0 \end{array}$$

$$\text{final output} = \bigcirc \rightarrow \boxed{a_F x + b_F y + c_F z + d_F} = 0$$



Linear mapping basics of MLPs

Multi-Layer Perceptron (MLP)

An MLP of L hidden layers takes an input $x \in \mathbb{R}^{n_0}$, and computes $x_l = g(W_l x_{l-1})$ at per layer, where g is the ReLU activation. We compactly write the MLP as a mapping $Tx = g(W_L \dots g(W_1 x))$.

Neuron Functionals

Any k^{th} neuron, $k \in \{1, \dots, n_l\}$, of an l^{th} layer of the MLP T specifies a functional defined as

$$a_{lk}(\boldsymbol{x}) = \pi_k \boldsymbol{g}(\boldsymbol{W}_l \boldsymbol{g}(\boldsymbol{W}_{l-1} \dots \boldsymbol{g}(\boldsymbol{W}_1 \boldsymbol{x})))$$

where π_k denotes an operator that projects onto the k^{th} coordinate.

All the neurons at layer l define a functional

$$\boldsymbol{a}_l(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{W}_l \boldsymbol{g}(\boldsymbol{W}_{l-1} \dots \boldsymbol{g}(\boldsymbol{W}_1 \boldsymbol{x})))$$

Region/Cell

*Hyperplanes specified by **neuron functionals** at layer l from a **hyperplane arrangement**, which partitions the feature space of layer $l-1$ into multiple linear regions/cells.*

Let $\mathcal{R}$ denote the set of all linear regions/cells in $\mathbb{R}^{n_0}$ that are possibly achieved by T.

Region-wise functionals and mappings of MLPs

State functionals

For a k^{th} neuron of an l^{th} layer of an MLP T, with $k \in \{1, \dots, n_l\}$ and $l \in \{1, \dots, L\}$, its state functional of neuron activation is defined as

$$s_{lk}(\boldsymbol{x}) = \begin{cases} 1 & \textit{if } a_{lk}(\boldsymbol{x}) > 0 \\ 0 & \textit{if } a_{lk}(\boldsymbol{x}) \leq 0 \end{cases}$$

which gives the state functional of layer l as

$$\boldsymbol{s}_l(\boldsymbol{x}) = [s_{l1}(\boldsymbol{x}), \dots, s_{ln_l}(\boldsymbol{x})]^\top$$

and the state functional of MLP T as

$$\boldsymbol{s}(\boldsymbol{x}) = [s_1(\boldsymbol{x})^\top, \dots, s_L(\boldsymbol{x})^\top]^\top$$

Since $s(x)$ is fixed for all x that fall in the same region $r \in \mathcal{R}$, we use $s(r) \in \{0,1\}^N$ to label this region, where N is the number of all hidden neurons.

Region-wise functionals and mappings of MLPs

Region-wise linear mappings

Given a ReLU based MLP T of L hidden layers, for any region/cell $r \in \mathcal{R}$, its associated linear mapping T^r is defined as

$$\boldsymbol{T}^r = \prod_{l=1}^{L} \boldsymbol{W}_l^r \qquad \boldsymbol{W}_l^r = diag(\boldsymbol{s}_l(r))\boldsymbol{W}_l$$

Neuron-wise linear mappings

With above definitions, we can give a linear mapping of a k^{th} neuron of layer l from input space.

$$\boldsymbol{a}_{lk}^r = \pi_k \prod_{i=1}^{l} \boldsymbol{W}_i^r$$

When MLPs are trained to be SDFs

Region-wise linear mappings

Given a ReLU based MLP T of L hidden layers, for any region/cell $r \in \mathcal{R}$, its associated linear mapping T^r is defined as

$$\boldsymbol{T}^r = \prod_{l=1}^{L} \boldsymbol{W}_l^r \qquad \boldsymbol{W}_l^r = diag(\boldsymbol{s}_l(r))\boldsymbol{W}_l$$

Functional of SDF

We stack on top of T a regression function $f\colon \mathbb{R}^{n_L} \to \mathbb{R}$, giving rise to a functional of SDF as

$$F(\boldsymbol{x}) = f \circ \boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{w}_f^\top \boldsymbol{g}(\boldsymbol{W}_L \ldots \boldsymbol{g}(\boldsymbol{W}_1 \boldsymbol{x}))$$

Neuron-wise linear mappings

With above definitions, we can give a linear mapping of a k^{th} neuron of layer l from input space, and the linear mapping of output neuron.

$$\boldsymbol{a}_{lk}^r = \pi_k \prod_{i=1}^{l} \boldsymbol{W}_i^r \qquad \boldsymbol{a}_F^r = \boldsymbol{w}_f^\top \boldsymbol{T}^r = \boldsymbol{w}_f^\top \prod_{i=1}^{L} \boldsymbol{W}_i^r$$

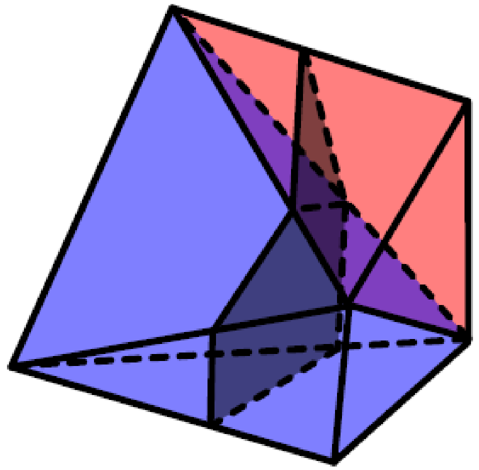
When MLPs are trained to be SDFs

Feasible region/cell

Let $\{\tilde{r} \in \tilde{\mathcal{R}}\}$ denote the subset of regions in $\mathcal{R}$ that have intersection with the zero-level isosurface.

For any $x \in \tilde{r}$, it must satisfy the following system of inequalities.

$$(\boldsymbol{I} - 2\text{diag}(\boldsymbol{s}(\tilde{r}))\boldsymbol{A}^{\tilde{r}}\boldsymbol{x} = \begin{bmatrix} (1-2s_{11}(\tilde{r}))\boldsymbol{a}_{11}^{\tilde{r}} \\ \vdots \\ (1-2s_{lk}(\tilde{r}))\boldsymbol{a}_{lk}^{\tilde{r}} \\ \vdots \\ (1-2s_{Ln_L}(\tilde{r}))\boldsymbol{a}_{Ln_L}^{\tilde{r}} \end{bmatrix} \boldsymbol{x} \preceq 0$$



Analytic cell (with zero-level isosurface)

When the region is bounded, the above inequalities essentially forms a polyhedral cell defined as

$$\mathcal{C}_F^{\tilde{r}} = \{\boldsymbol{x} \in \mathbb{R}^3 | (\boldsymbol{I} - 2\text{diag}(\boldsymbol{s}(\tilde{r}))\boldsymbol{A}^{\tilde{r}}\boldsymbol{x} \preceq 0\}$$

Analytic face

We define the polygonal face that is an intersection of analytic cell $\tilde{r}$ and surface $\mathcal{Z}$.

$$\mathcal{P}_F^{\tilde{r}} = \{\boldsymbol{x} \in \mathbb{R}^3 | \boldsymbol{a}_F^{\tilde{r}}\boldsymbol{x} = 0, (\boldsymbol{I} - 2\text{diag}(\boldsymbol{s}(\tilde{r}))\boldsymbol{A}^{\tilde{r}}\boldsymbol{x} \preceq 0\}$$

A theorem that guarantees analytic cells/faces connect to form a closed mesh

✓ We have the following theorem that characterizes the (mild) conditions under which analytic cells and faces are guaranteed to connect and form a closed, piecewise planar surface.

Theorem 1. Assume that the zero-level isosurface $\mathcal{Z}$ of a SDF $F = f \circ \boldsymbol{T}$ defines a closed, piecewise planar surface. If for any region/cell $r \in \mathcal{R}(\boldsymbol{T})$, its associated linear mapping $\boldsymbol{T}^r$ and the induced plane $\boldsymbol{a}_F^r = \boldsymbol{w}_f^\top \boldsymbol{T}^r$ are uniquely defined, i.e. $\boldsymbol{T}^r \neq \beta \boldsymbol{T}^{r'}$ and $\boldsymbol{a}_F^r \neq \beta \boldsymbol{a}_F^{r'}$ for any region pair of r and r', where β is an arbitrary scaling factor, then analytic faces $\{\mathcal{P}_F^{\tilde{r}}\}$ connect and exactly form the surface $\mathcal{Z}$.

✓ ***Remarks***

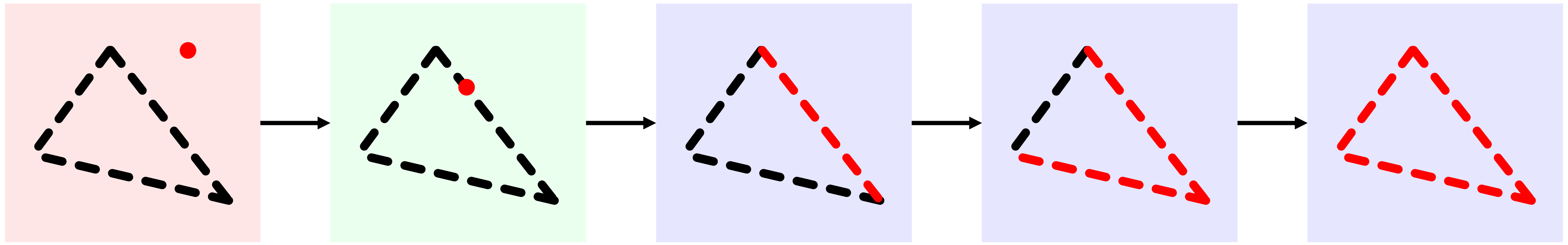
The conditions assumed are practically reasonable up to a numerical precision of the learned network weights in the SDF $F = f \circ \boldsymbol{T}$.

Proof of the theorem also suggests an algorithm to identify the polygonal faces of a surface mesh learned by F, which is to be presented shortly.

An algorithm of Analytic Marching

Overview

Given a learned SDF $F = f \circ T$ whose zero-level isosurface defines a closed, piecewise planar surface, a theorem proved by us suggests that obtaining the mesh concerns with identification of analytic faces $\mathcal{P}_F^{\tilde{r}}$ in analytic cell $\{\mathcal{C}_F^{\tilde{r}} | \tilde{r} \in \tilde{\mathcal{R}}\}$. To this end, we propose an algorithm of Analytic Marching that marches among analytic cells $\{\mathcal{C}_F^{\tilde{r}} | \tilde{r} \in \tilde{\mathcal{R}}\}$ to identify vertices and edges of the polygonal faces $\{\mathcal{P}_F^{\tilde{r}} | \tilde{r} \in \tilde{\mathcal{R}}\}$.



① Randomly initialize one point in 3D space

② Move the point to the surface via simple gradient descent

③ Identify one analytic face via solving 3x3 systems of linear inequalities

④ March to the next, neighboring analytic cell via transition of neuron state

⑤ Repeat the above until an end condition of no analytic cells identified

Analytic Marching – algorithmic details

The steps of Analytic Marching are as follows:

1. Identify at least one point $\boldsymbol{x} \in \mathcal{Z}$.
   Given the parametric model F and an arbitrarily initialized point $x \in \mathbb{R}^3$, this can be simply achieved by solving the following problem with stochastic gradient descent (SGD).

$$\min_{\boldsymbol{x} \in \mathbb{R}^3} |F(\boldsymbol{x})|$$

2. Compute state vector $s(x)$, and initialize an active set $\mathcal{S}^{\bullet} = \emptyset$ and an inactive set $\mathcal{S}^{\circ} = \emptyset$, push $s(x)$ into $\mathcal{S}^{\bullet}$.
3. Take an active state s_i from $\mathcal{S}^{\bullet}$, which specifies its analytic cell $\mathcal{C}_F^{\tilde{r}_i}$ and analytic face $\mathcal{P}_F^{\tilde{r}_i}$.
4. Enumerate all the vertices of the analytic face defined by

$$\mathcal{P}_F^{\tilde{r}_i} = \{\boldsymbol{x} \in \mathbb{R}^3 | \boldsymbol{a}_F^{\tilde{r}_i} \boldsymbol{x} = 0, (\boldsymbol{I} - 2\text{diag}(\boldsymbol{s}(\tilde{r}_i))\boldsymbol{A}^{\tilde{r}_i}\boldsymbol{x} \preceq 0\}$$

   Note that it can be done simply by brute-force vertices enumeration.
   Record all the boundary planes $\{\widehat{H}_{lk}^{\tilde{r}_i}\}$ of $\mathcal{C}_F^{\tilde{r}_i}$ that give valid vertices.
5. Push s_i out of the active set $\mathcal{S}^{\bullet}$, and into the inactive set $\mathcal{S}^{\circ}$.
6. Infer the state vector $\widehat{s}_i$ of adjacent faces, and push $\{\widehat{s}_i | \widehat{s}_i \notin \mathcal{S}^{\circ}\}$ into the active set $\mathcal{S}^{\bullet}$.
   Note that the analytic cell connecting $\mathcal{C}_F^{\tilde{r}_i}$ at a boundary $\widehat{H}_{lk}^{\tilde{r}_i}$ has its state vector switching only at the k^{th} neuron of layer l.
7. Repeat steps 3 to 6, until the active set $\mathcal{S}^{\bullet}$ is cleared up.

Analytic Marching – computational analysis and parallel implementation

Computational complexities

Assume that the SDF $F = f \circ T$ is built on an MLP of L hidden layers, each of which has n neurons, and thus $N = nL$

The complexity of our Analytic Marching algorithm has an order of $\mathcal{O}\left((n/2)^{2(L-1)}|\mathcal{V}_{\mathcal{P}}|n^4L^2\right)$

Ideally, its complexity is exponential w.r.t. depth L and polynomial w.r.t. width n

Parallel implementations

The proposed analytic marching can be naturally implemented in parallel, and initialization of multiple surface points would help recover components of the surface that are possibly isolated

Training objective

For any $x \in \mathbb{R}^3$, let $d(x)$ denote its ground-truth value of signed distance to the surface. We use the following regularized objective to train a SDF $F = f \circ T$,

$$\min_{F=f\circ \boldsymbol{T}} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{R}^3}\left|F(\boldsymbol{x})-d(\boldsymbol{x})\right|+\alpha\left|\|\partial F(\boldsymbol{x})/\partial\boldsymbol{x}\|-1\right|$$

The unit gradient regularizer aims to promote learning of a smooth gradient field

Experiments

Dataset

ShapeNet - A repository of many 3D shapes

Evaluation metrics

Chamfer Distance (CD), Earth Movers Distance (EMD),

Intersection over Union (IoU), F-score (F),

Wall-clock time (Time), Number of faces (Face No.).

Ablation study – Depth and Width

Architecture	CD($\times 10^{-1}$)	EMD($\times 10^{-3}$)	IoU(%)	F@τ(%)	F@2τ(%)	Face No.	Time(sec.)
D4-W90	6.16	8.60	86.5	84.8	95.4	97865	17.0
D6-W60	5.29	8.00	86.9	85.5	95.8	100179	15.0
D8-W45	5.67	8.12	85.7	84.2	95.4	93482	9.75
D10-W90	3.64	5.85	90.2	88.1	98.6	421840	173
D15-W60	3.85	6.23	88.9	87.4	97.5	336044	86.8
D20-W45	4.21	6.89	86.1	86.6	95.5	253970	47.3

“D” stands for depth, “W” stands for width.

Given a fixed number of neurons, it seems that properly deep networks are advantageous in terms of precision-efficiency trade off

Experiments

Results of other object categories

It tells that different categories have different complexity. It seems that the complexities of “Chair” are higher, and those of “Airplane” are lower.

Category	CD($\times 10^{-1}$)	EMD($\times 10^{-3}$)	IoU(%)	F@τ(%)	F@2τ(%)	Face No.	Time(sec.)
Rifle	5.29	8.00	86.9	85.5	95.8	100179	15.0
Chair	7.27	8.01	89.6	52.9	96.1	182624	25.3
Airplane	3.25	5.32	89.4	89.8	97.9	131074	19.0
Sofa	6.78	6.65	96.6	53.2	97.9	156863	22.0
Table	6.98	7.16	90.7	51.6	96.8	163395	22.7
Mean	5.91	7.03	90.6	66.6	96.9	146827	20.8

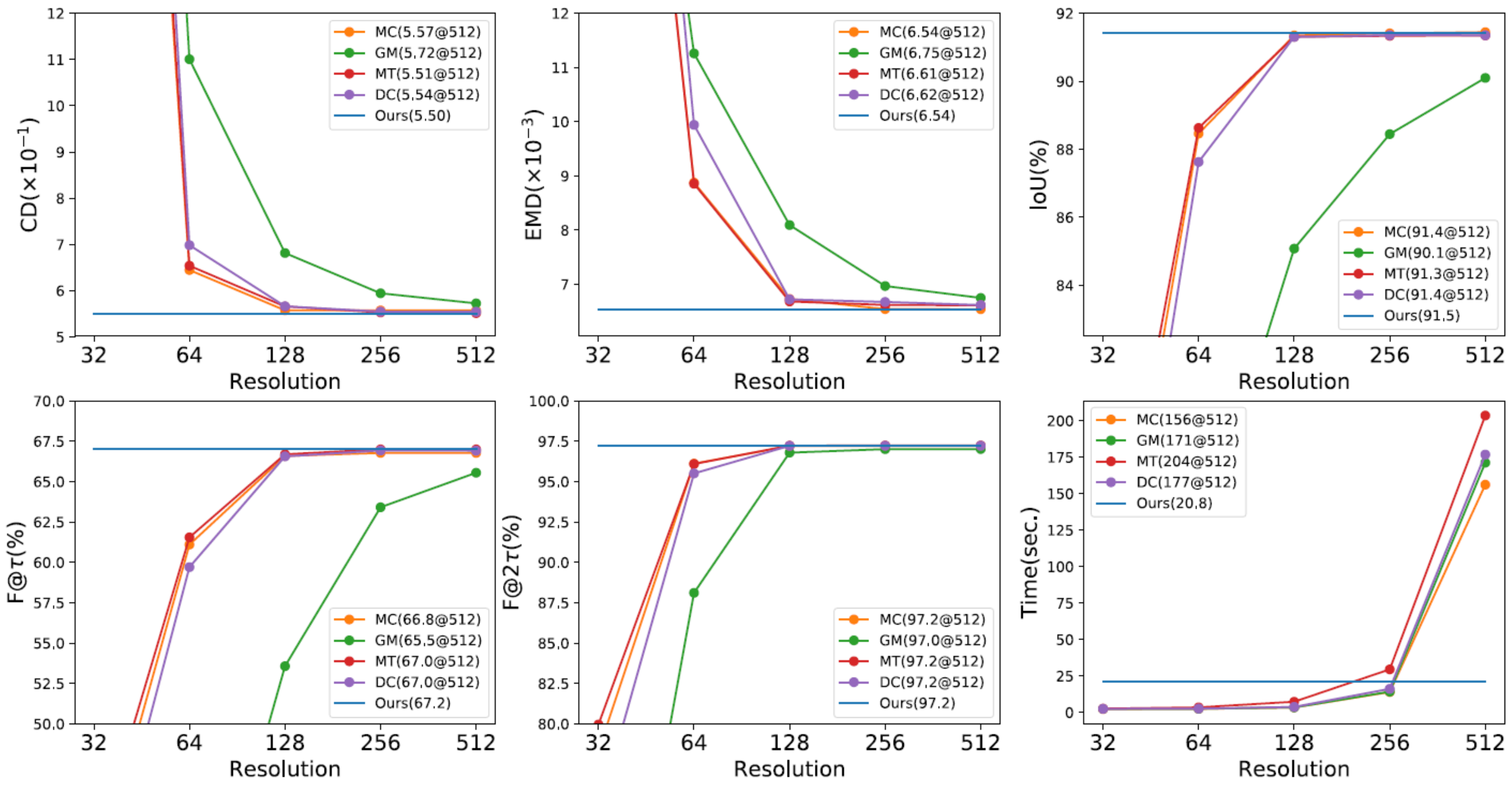
Comparisons with existing meshing algorithms

We compare our proposed algorithm (Analytic Marching, AM) with existing algorithms of Greedy Meshing (GM), Marching Cubes (MC), Marching Tetrahedra (MT), and Dual Contouring (DC).

Their meshing accuracy depends on the sampling resolution. We implement them under a range of sampling resolutions from 32^3 to a GPU memory limit of 512^3

Experiments

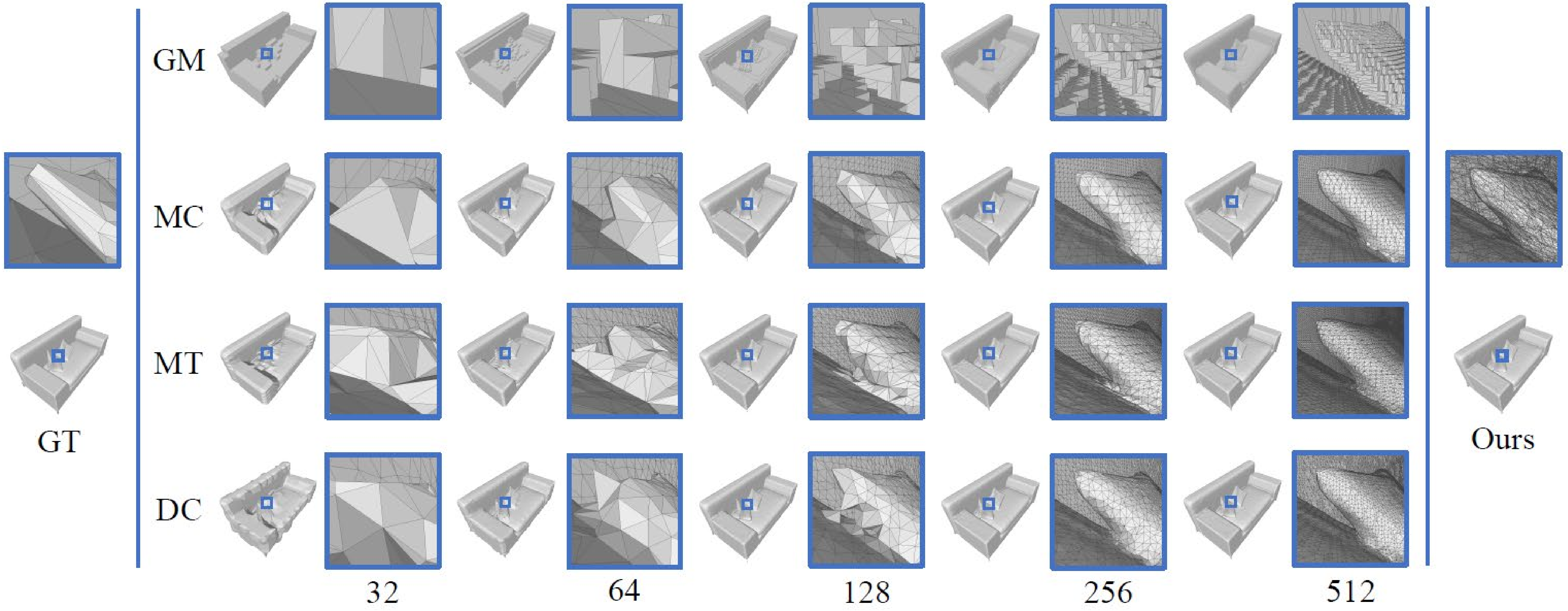
Comparisons with existing meshing algorithms



Under different evaluation metrics, recovery accuracies of these methods are upper bounded by our proposed one. Note that the dominating computations of competitors are implemented on GPU, which gives them an unfair advantage of computational efficiency.

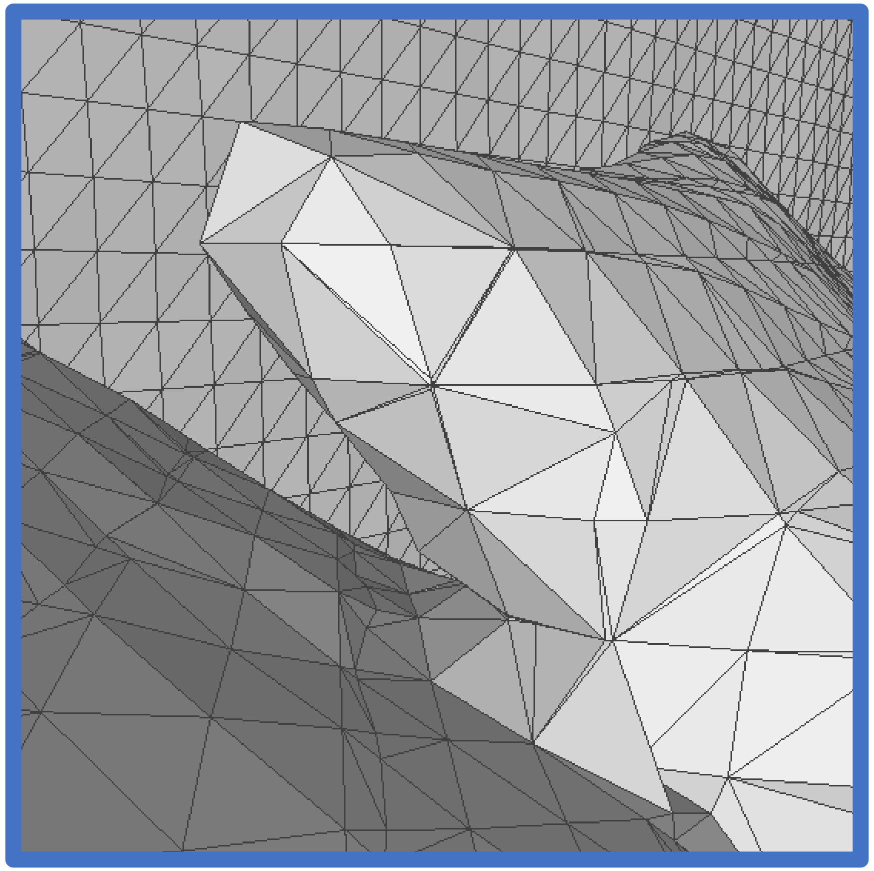
Experiments

Comparisons with existing meshing algorithms

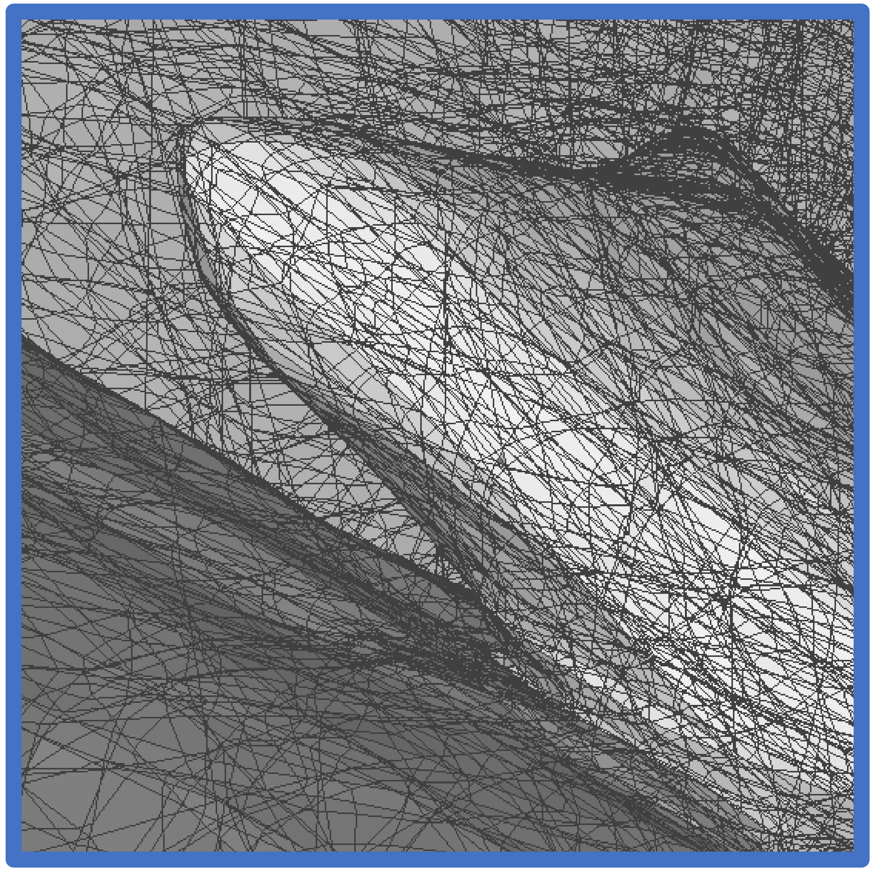


Experiments

Comparisons with existing meshing algorithms



MC128



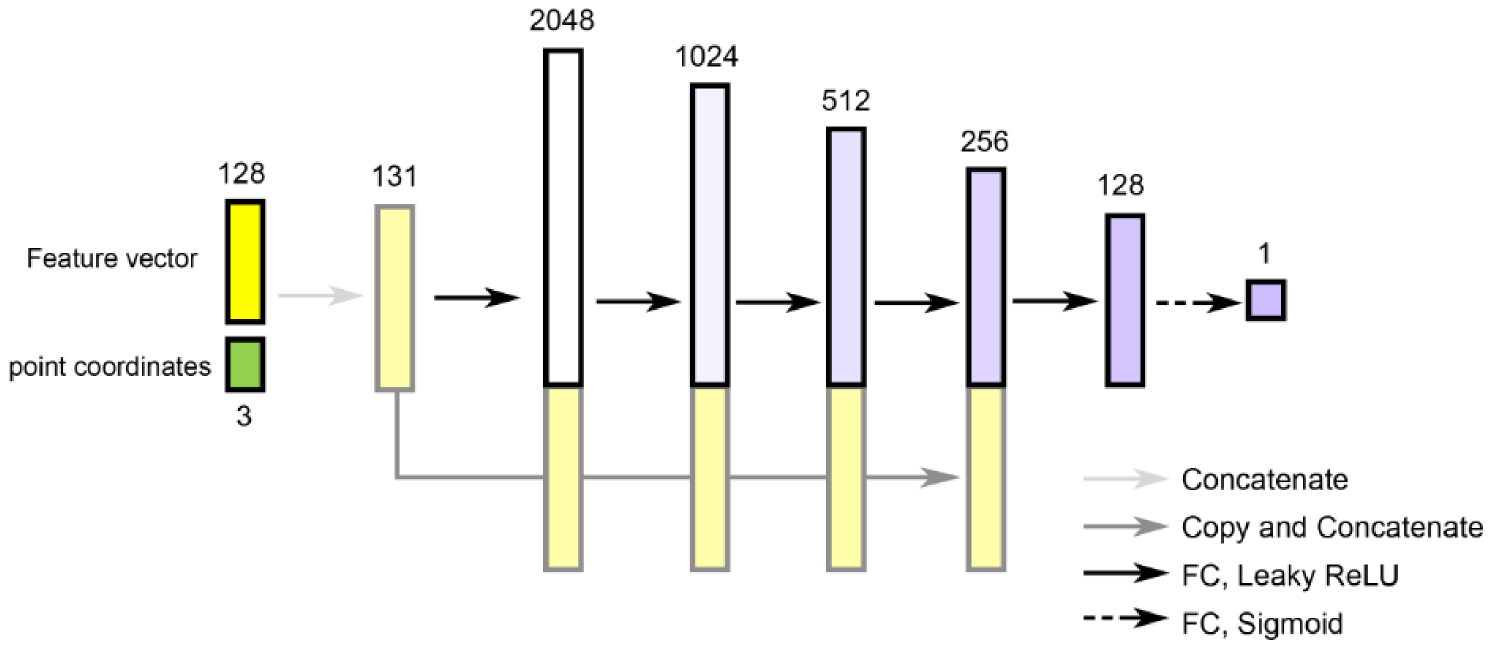
AM

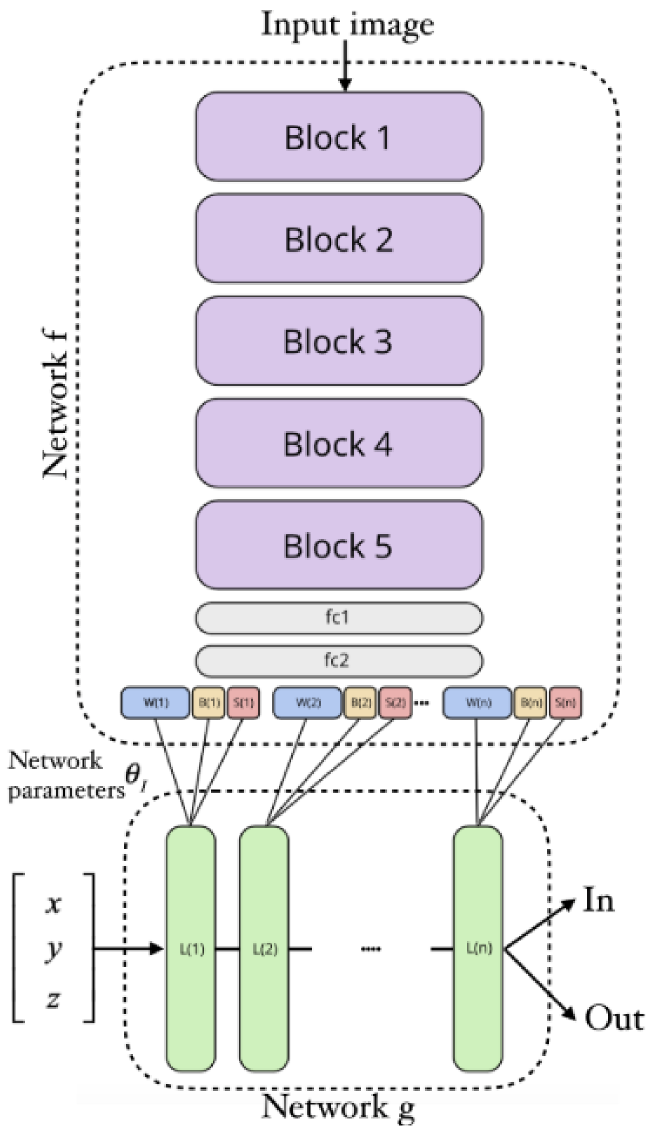
Experiments

The 3D mesh obtained by ours can be mathematically guaranteed to be a polygonal mesh

Promising directions

- ✓ Use hypernetwork to learn parameters of the MLP
- ✓ Apply postprocessing to retain fine details of the surface mesh
- ✓ Other explorations of the relationship between deep learning and shape representation
- ✓ A more efficient implementation of Analytic Marching





Thank you!