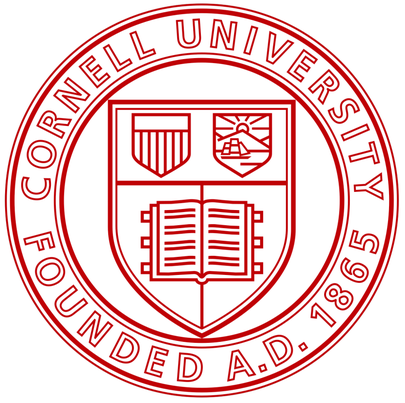
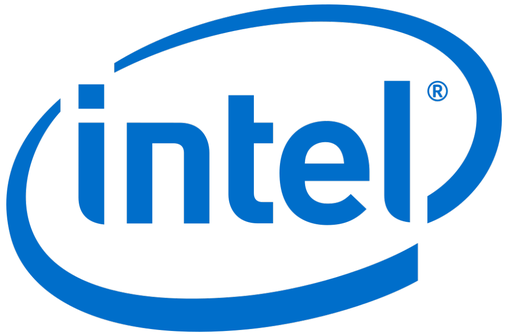
Geometry Processing with Neural Fields

Guandao Yang, Serge Belongie, Bharath Hariharan, Vladlen Koltun

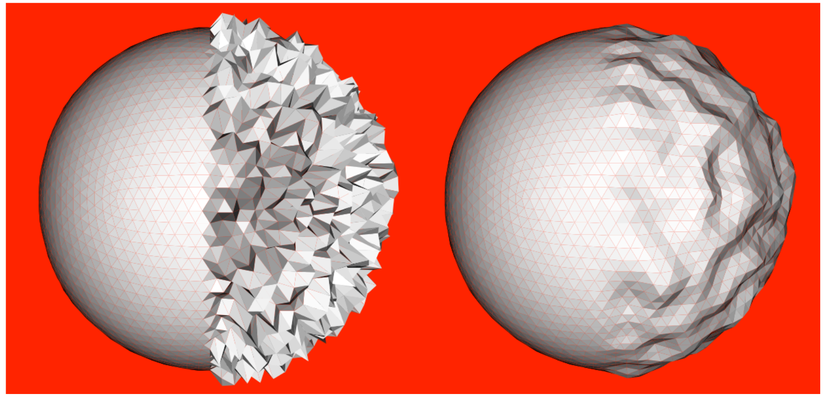




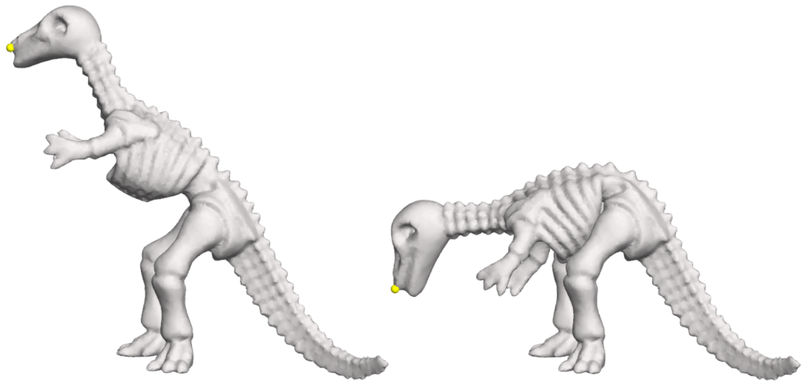
Geometry Processing
- manipulation of geometry data

Geometry Data

Geometry Processing Tasks



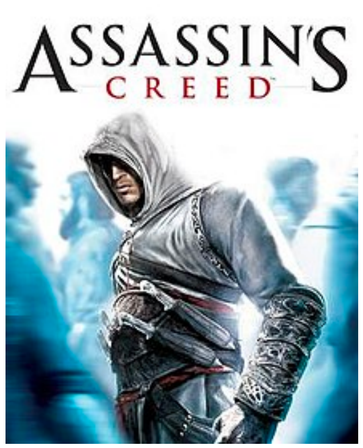
(Gabriel Taubin, 1995)



(Sorkine and Alexa, 2007)

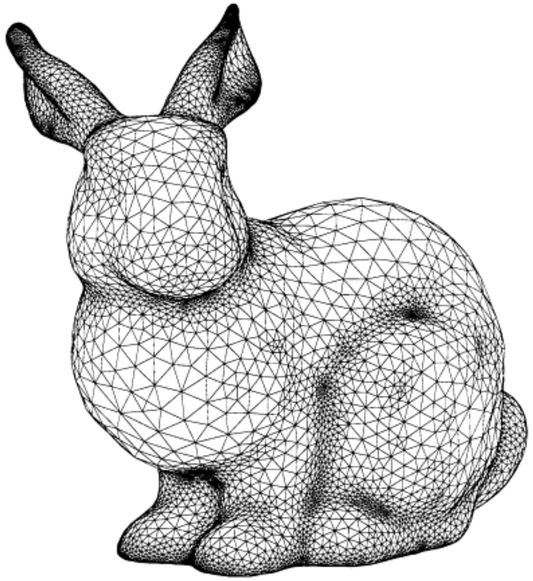
Applications

Computer Aided Design



Gaming and Movies

Representing Geometry Data



$$V = \{(x_i, y_i, z_i)\}_{i=1}^{n}$$
$$F = \{(u, v, w) \mid 1 \leq u, v, w, \leq n\}$$

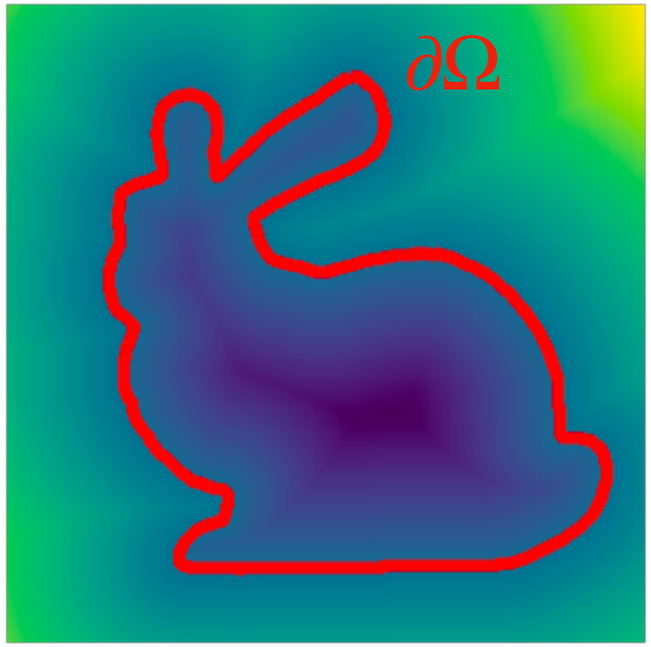
✓ **Easy to understand**

✓ **Fast access to local neighbor**

✗ **Topological changes**

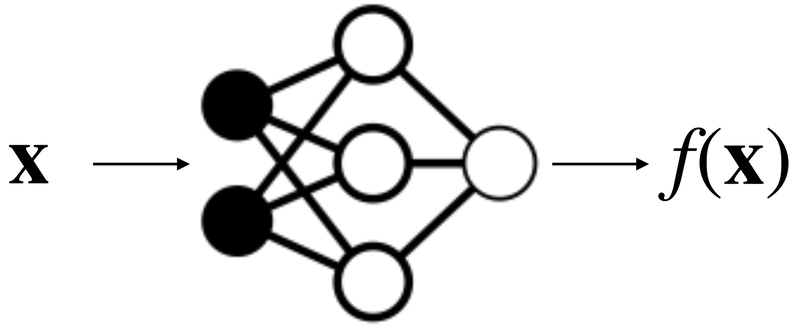
✗ **Need to maintain good surface discretization**

Representing Geometry Data



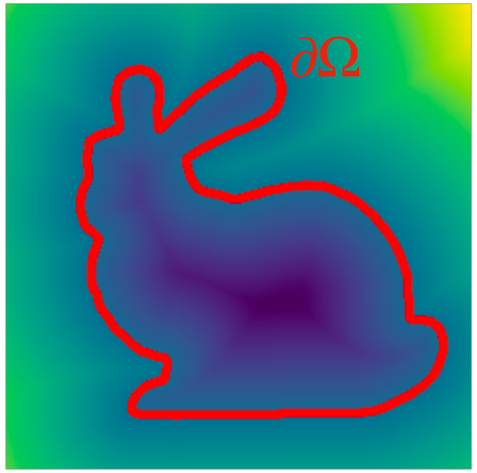
$$f : \mathbb{R}^3 \rightarrow \mathbb{R}$$

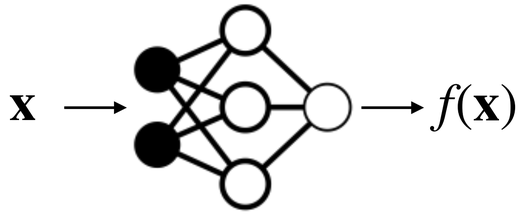
$$\partial\Omega = \{\mathbf{x} \mid f(\mathbf{x}) = c\}$$



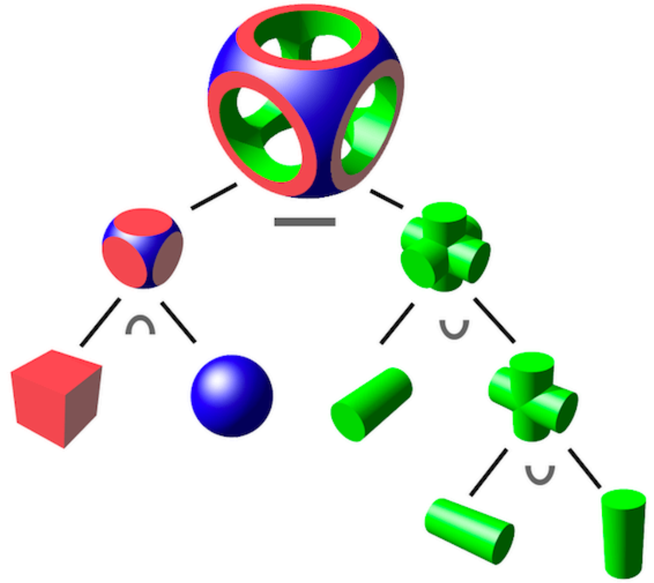
Neural Fields

Representing Geometry Data



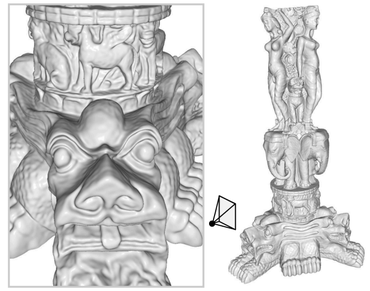


Neural Fields



✓ **Topological changes**

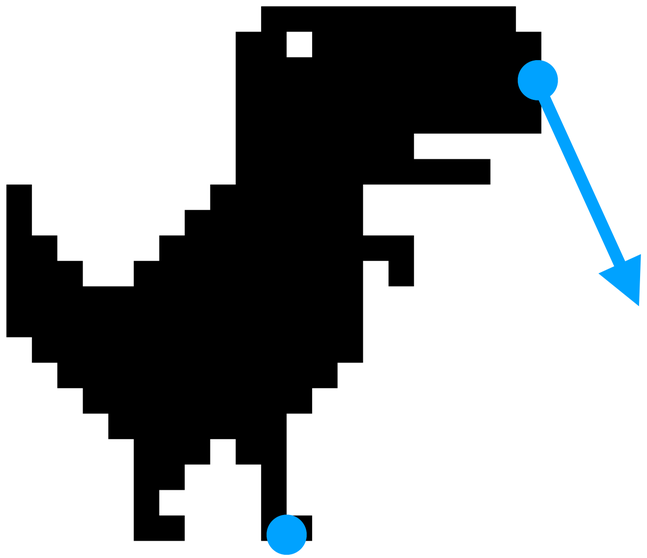
✓ **Compact**



✓ **High resolution**

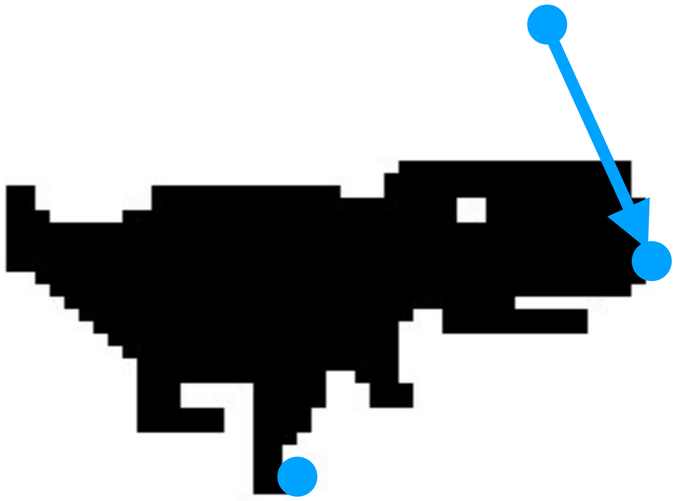
Can geometry processing be done entirely using Neural Fields?

Can geometry processing be done entirely using Neural Fields?



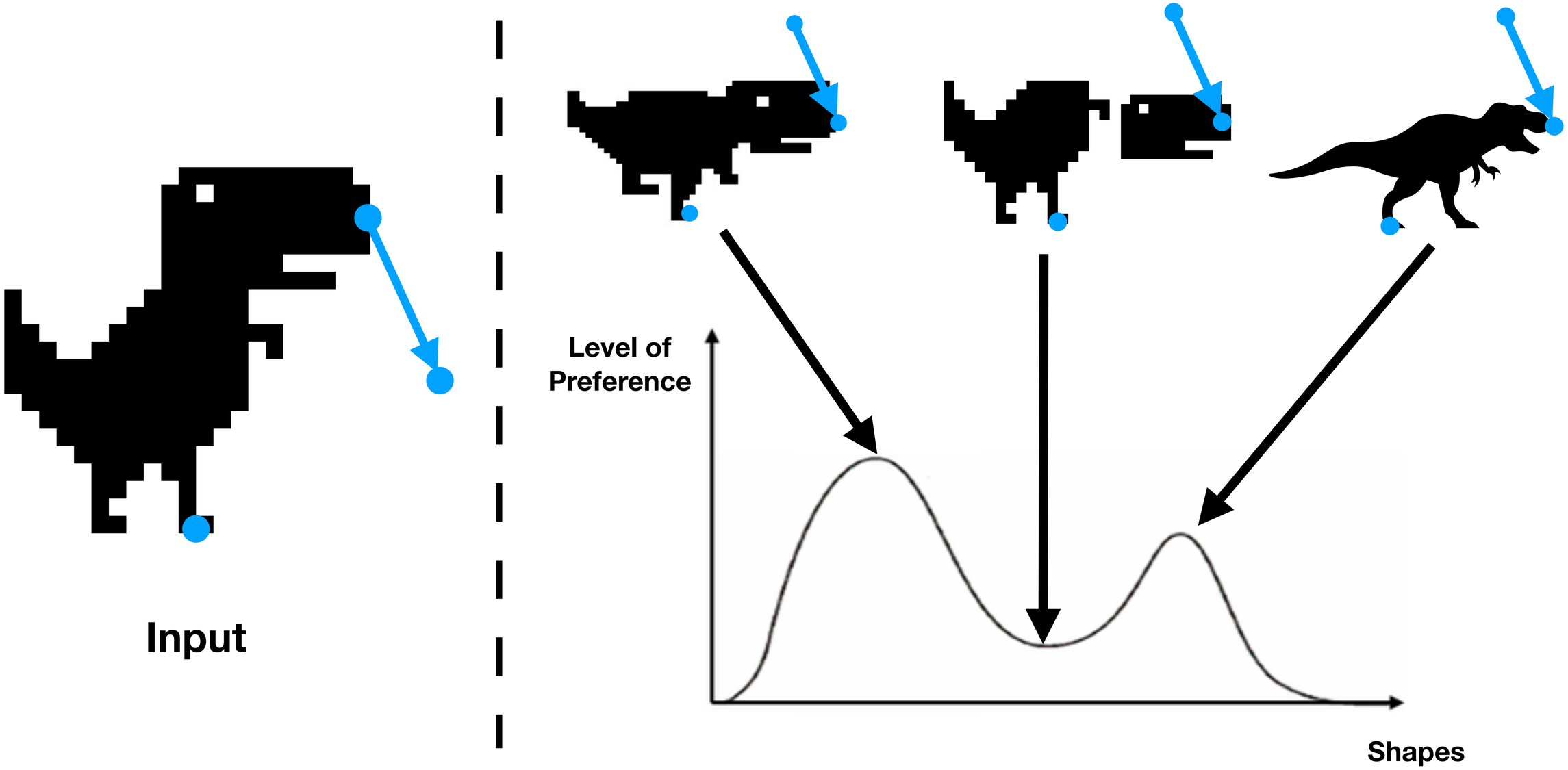
Input Shape

User Specification



Desirable Output

Geometry Processing is under-constrained

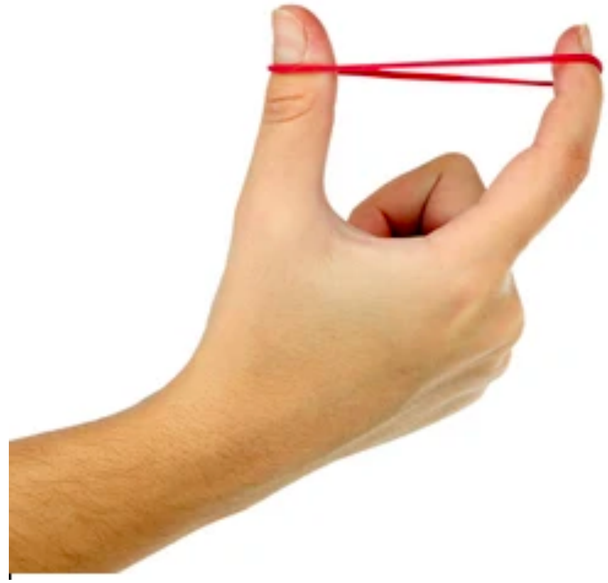


Physics-inspired Prior

Smooth

Kazhdan *et. al.*, 2006;
Kazhdan and Hoppe, 2013

Bending Resistance



Stretching Resistance

Terzopoulos et. al., 1987;
Sorkine and Alexa, 2007;
Levi and Gotsman, 2015

Quantify Priors

$$f : (u, v) \mapsto (x, y, z)$$

Normal

$$\mathbf{n} = \frac{f_u \times f_v}{\| f_u \times f_v \|}$$

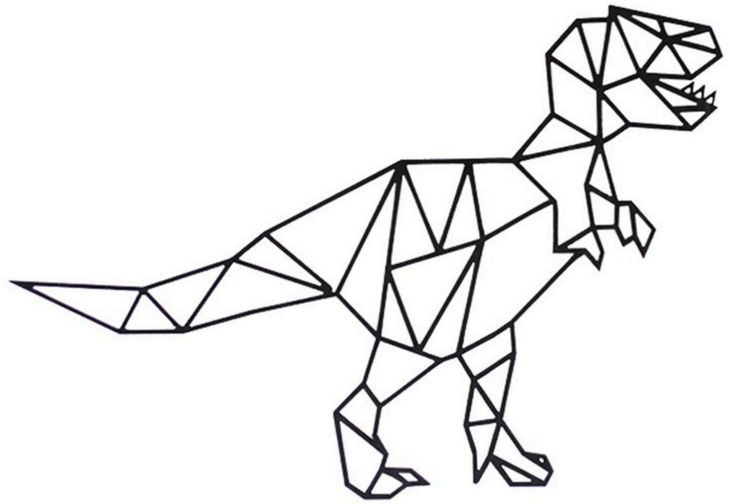
Area

$$\mathbf{I} = \begin{bmatrix} f_u^T f_u & f_u^T f_v \\ f_v^T f_u & f_v^T f_v \end{bmatrix} \quad S = \iint |\mathbf{I}|^{\frac{1}{2}} \, dudv$$

Curvature

$$\mathbf{II} = \begin{bmatrix} f_{uu}^T \mathbf{n} & f_{uv}^T \mathbf{n} \\ f_{vu}^T \mathbf{n} & f_{vv}^T \mathbf{n} \end{bmatrix} \quad \kappa = \frac{|\mathbf{II}|}{|\mathbf{I}|}$$

Quantify Prior - Mesh



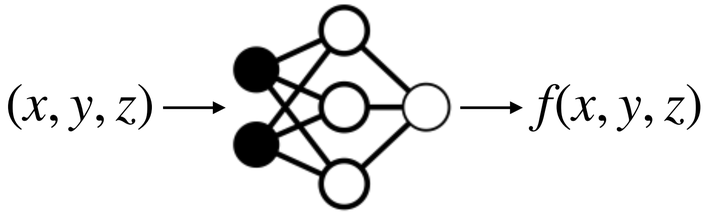
$$V = \{(x_i, y_i, z_i)\}_{i=1}^{n}$$
$$F = \{(u, v, w) \,|\, 1 \leq u, v, w, \leq n\}$$

Normal $$\mathbf{n}_{i,j,k} = \frac{(V_j - V_i) \times (V_k - V_i)}{|(V_j - V_i) \times (V_k - V_i)|}$$

Area $$S = \sum_{(i,j,k) \in F} \frac{1}{2} \left| (V_j - V_i) \times (V_k - V_i) \right|$$

Curvature $$\kappa_i = \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + cot\beta_{ij})(V_i - V_j)$$

Quantify Prior - Neural Fields



$$\partial\Omega = \{(x, y, z) \mid f(x, y, z) = 0\}$$

Normal

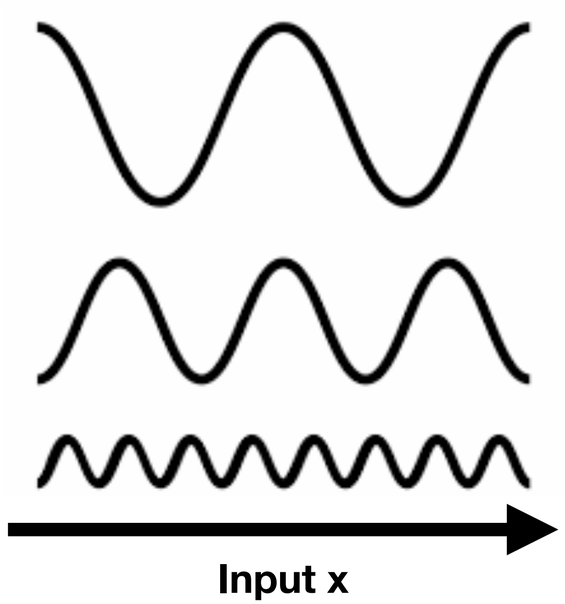
Area

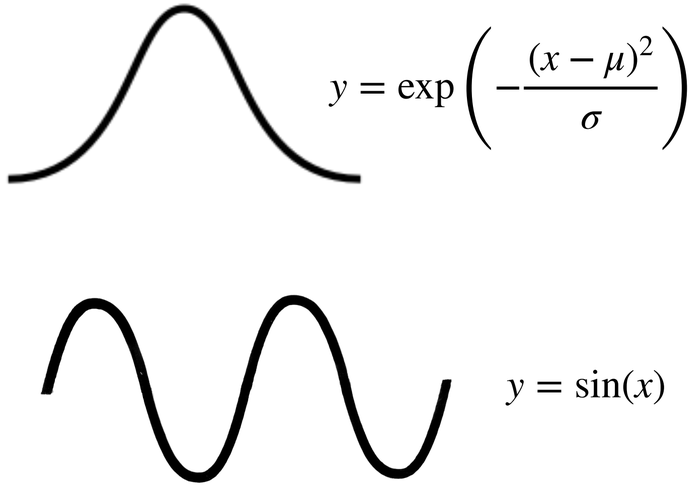
Curvature

Differentiability of Neural Fields

PyTorch

TensorFlow



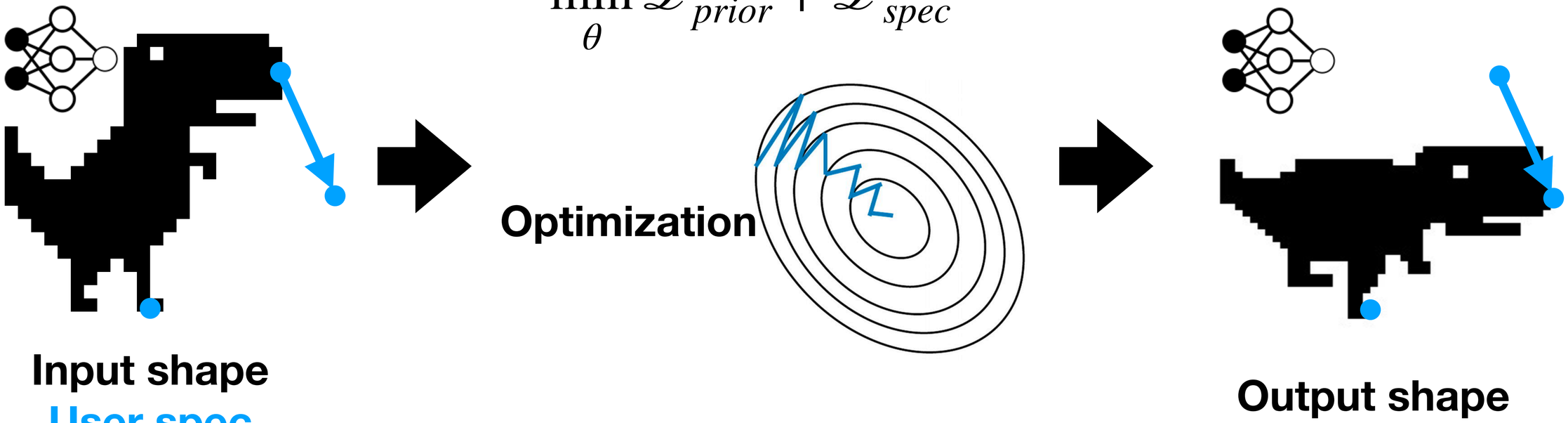


AutoGrad Frameworks

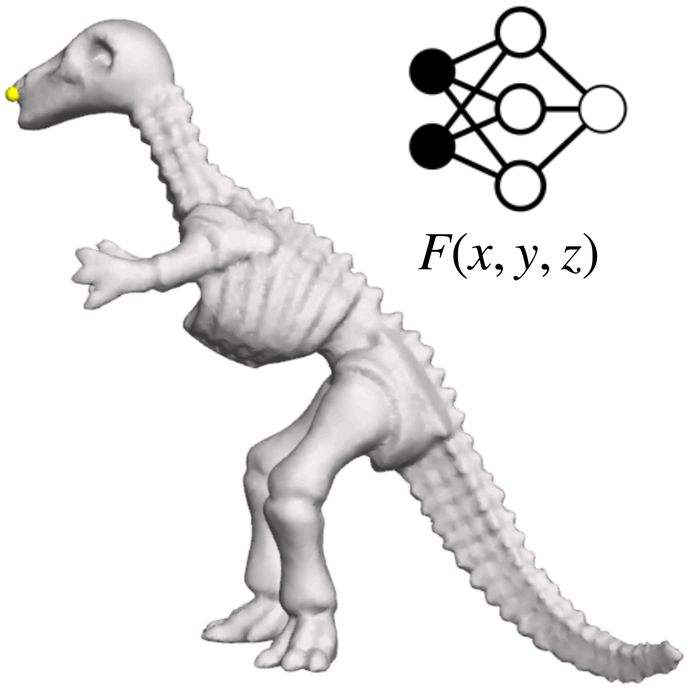
Positional Encoding

Smooth Activation

Geometry Processing with Neural Fields



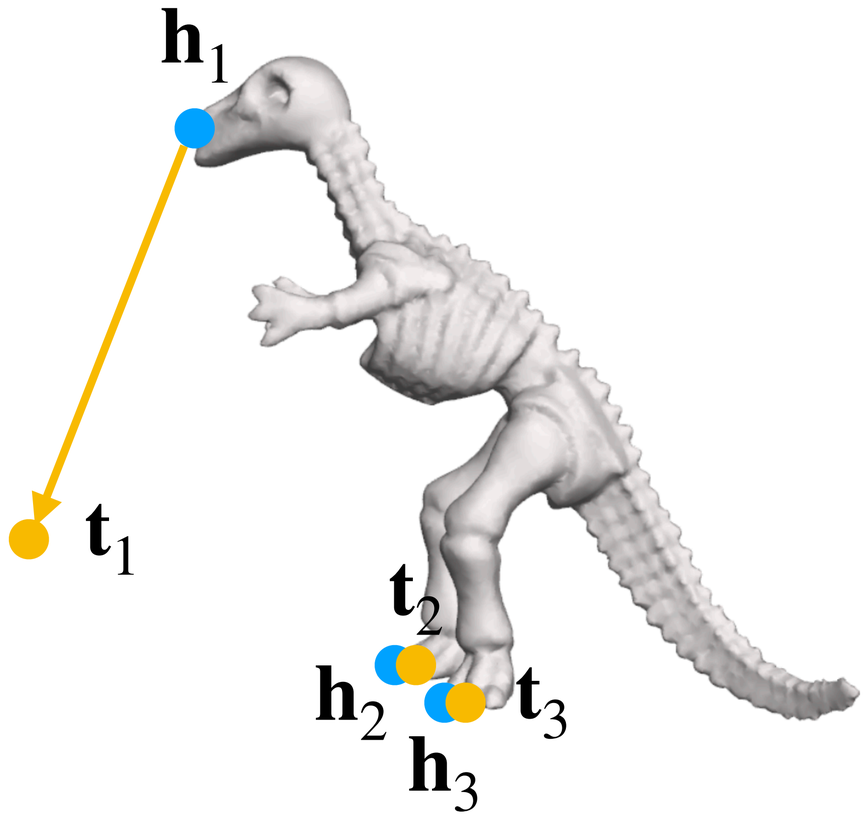
Deformation with Neural Fields



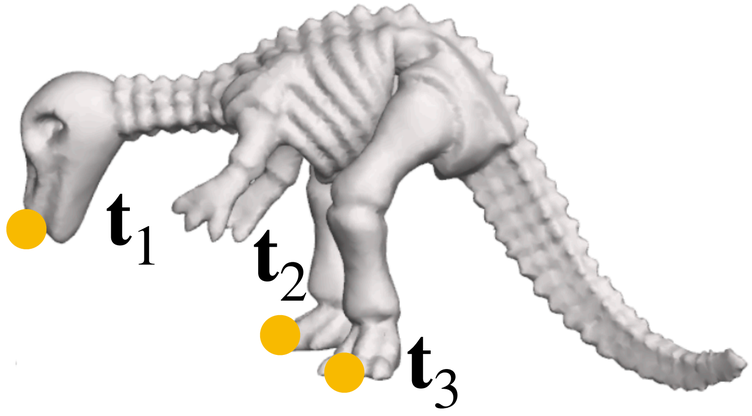
$F(x, y, z)$

$$F(\mathbf{x}) \approx \sigma(\mathbf{x}) \min_{\mathbf{p} \in \partial\Omega} \|\mathbf{p} - \mathbf{x}\|$$

$$\sigma(\mathbf{x}) = \begin{cases} -1 & \textbf{if } \mathbf{x} \in \Omega \\ 1 & \textbf{otherwise} \end{cases}$$



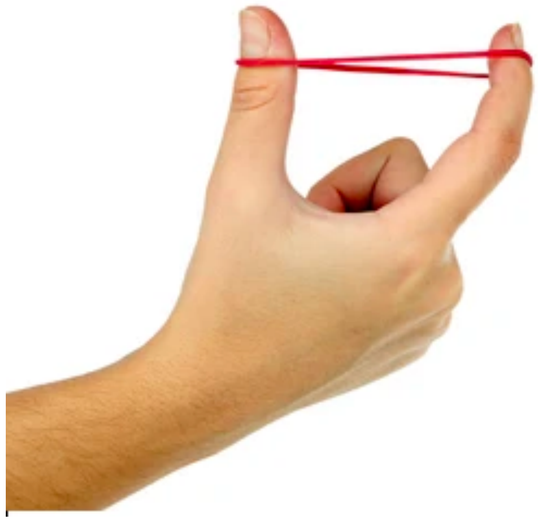
User specification



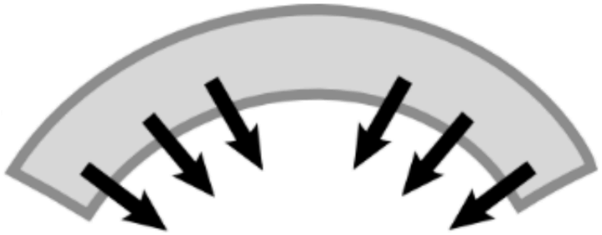
Desired Output

Natural Deformation

Elastic Deformation

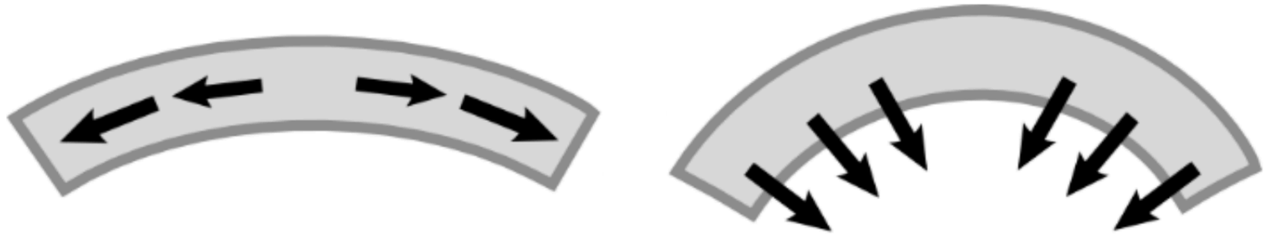


Stretching



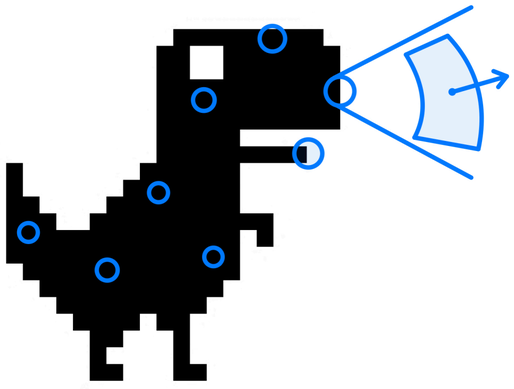
Bending

Elastic Deformation



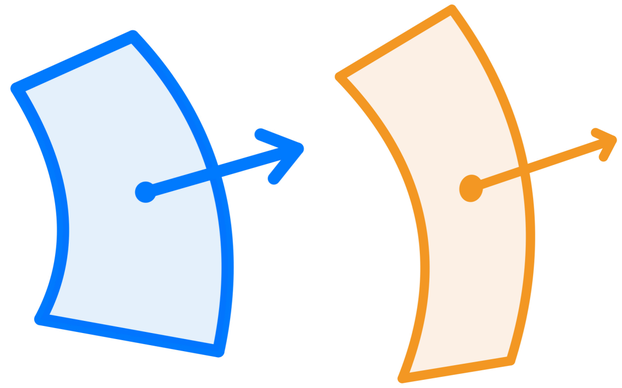
Stretching

Bending



1. Sampling

2. Correspondences



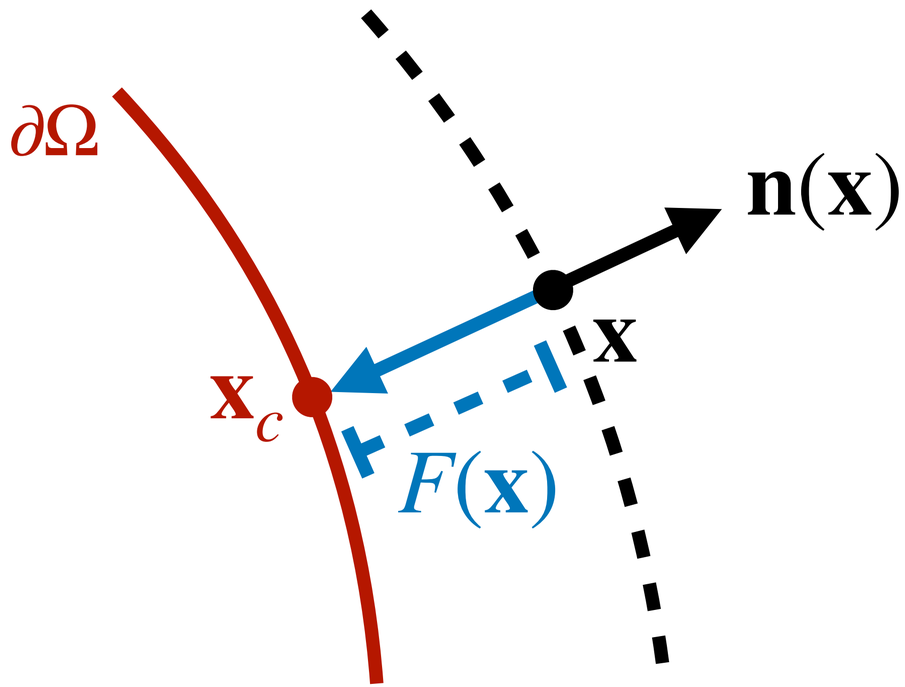
3. Comparing

Step 1: Sampling

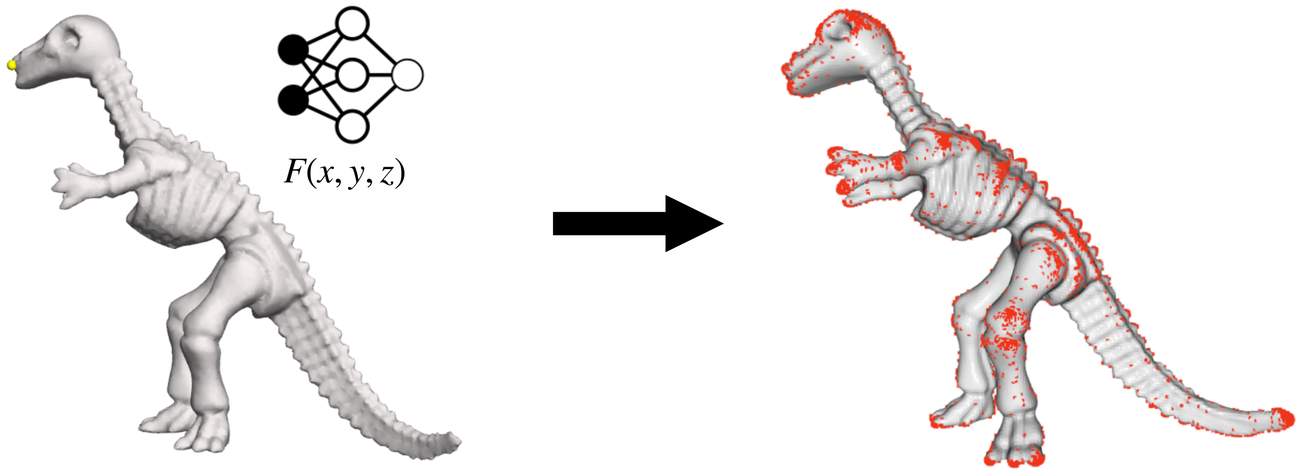
$$F(\mathbf{x}) \approx SDF(\mathbf{x})$$

$$\mathbf{x}_c = arg \min_{\mathbf{p} \in \partial\Omega} |\mathbf{p} - \mathbf{x}|$$

$$= \mathbf{x} - F(\mathbf{x})\mathbf{n}(\mathbf{x})$$



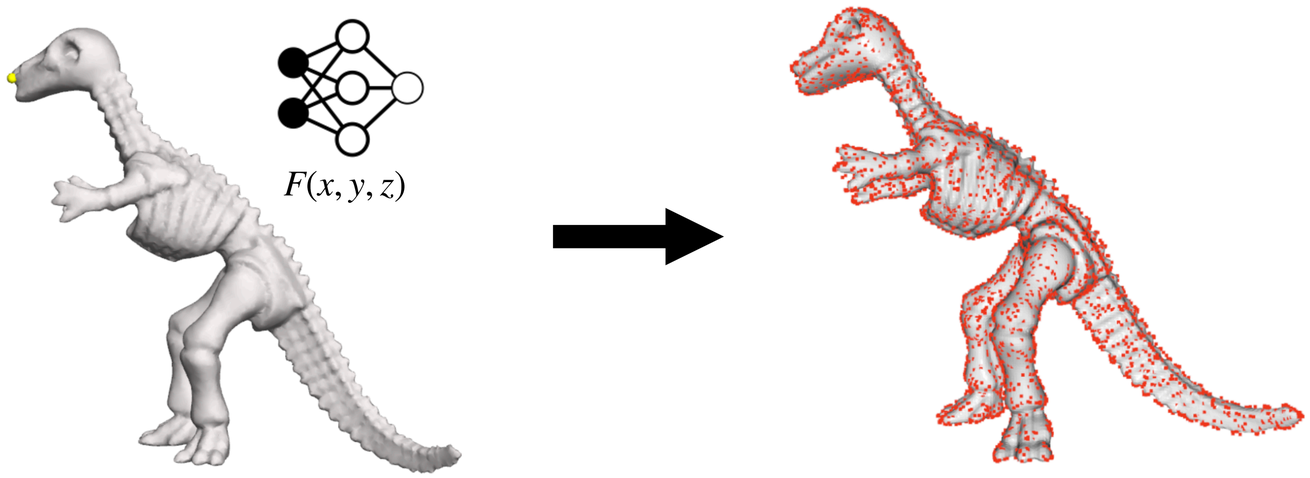
Step 1: Sampling



$$\mathbf{x}_0 \sim U([-1,1]^3)$$

$$\hat{\mathbf{x}} = \mathbf{x}_0 - F(\mathbf{x}_0)\mathbf{n}(\mathbf{x}_0)$$

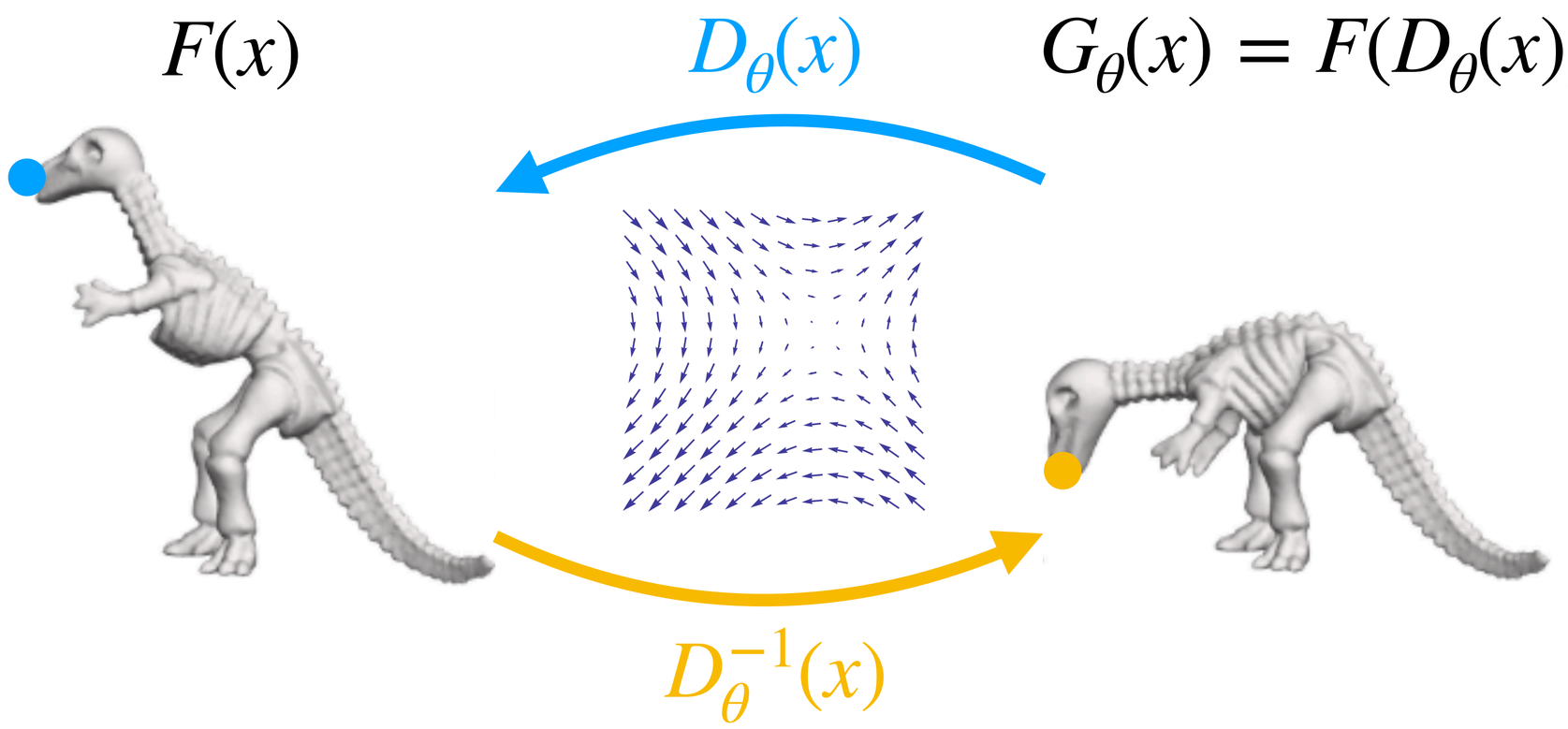
Step 1: Sampling



$$\mathbf{x}_0 \sim \{\mathbf{x} \,|\, \mathbf{x} \in U([-1,1]^3),\ F(\mathbf{x}) < \tau\}$$

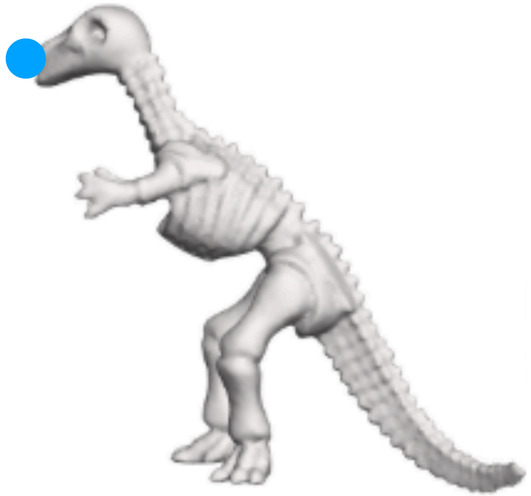
$$\hat{\mathbf{x}} = \mathbf{x}_0 - F(\mathbf{x}_0)\mathbf{n}(\mathbf{x}_0)$$

Step 2: Correspondences

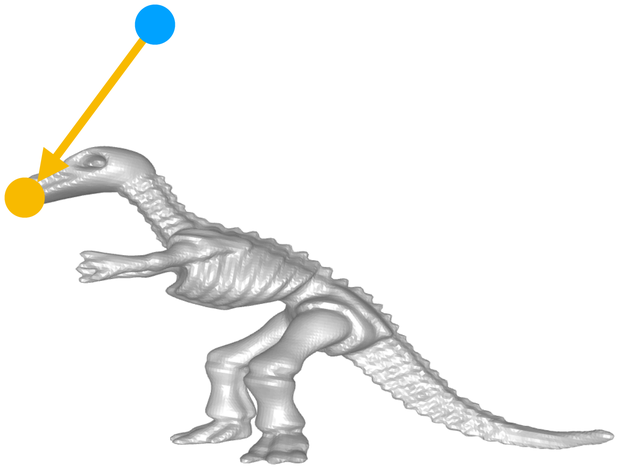


Step 2: Correspondences

$F(x)$ $D_\theta(x) = x + g_\theta(x)$ $G_\theta(x) = F(D_\theta(x))$

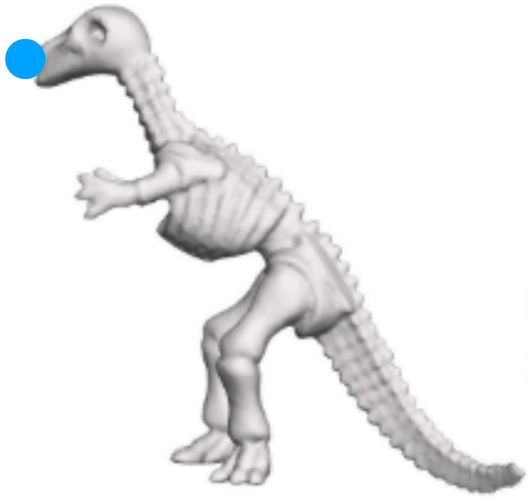


Invertible ResNet
(Behrmann *et. al.*, 2019)

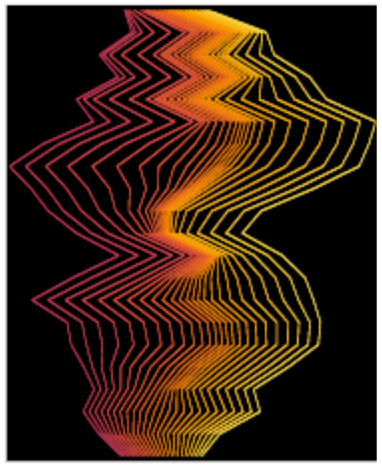


Step 2: Correspondences

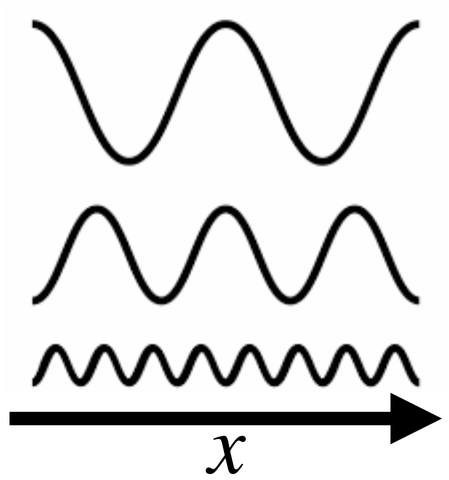
$F(x)$



$D_\theta(x) = x + g_\theta(x)$

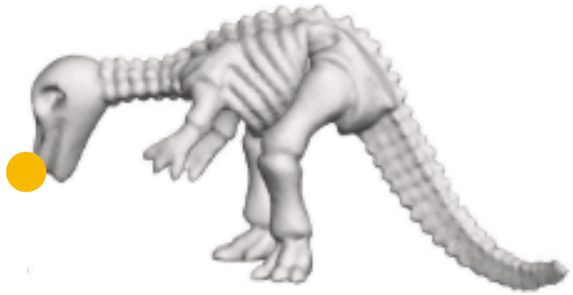


Invertible ResNet
(Behrman et. al., 2019)

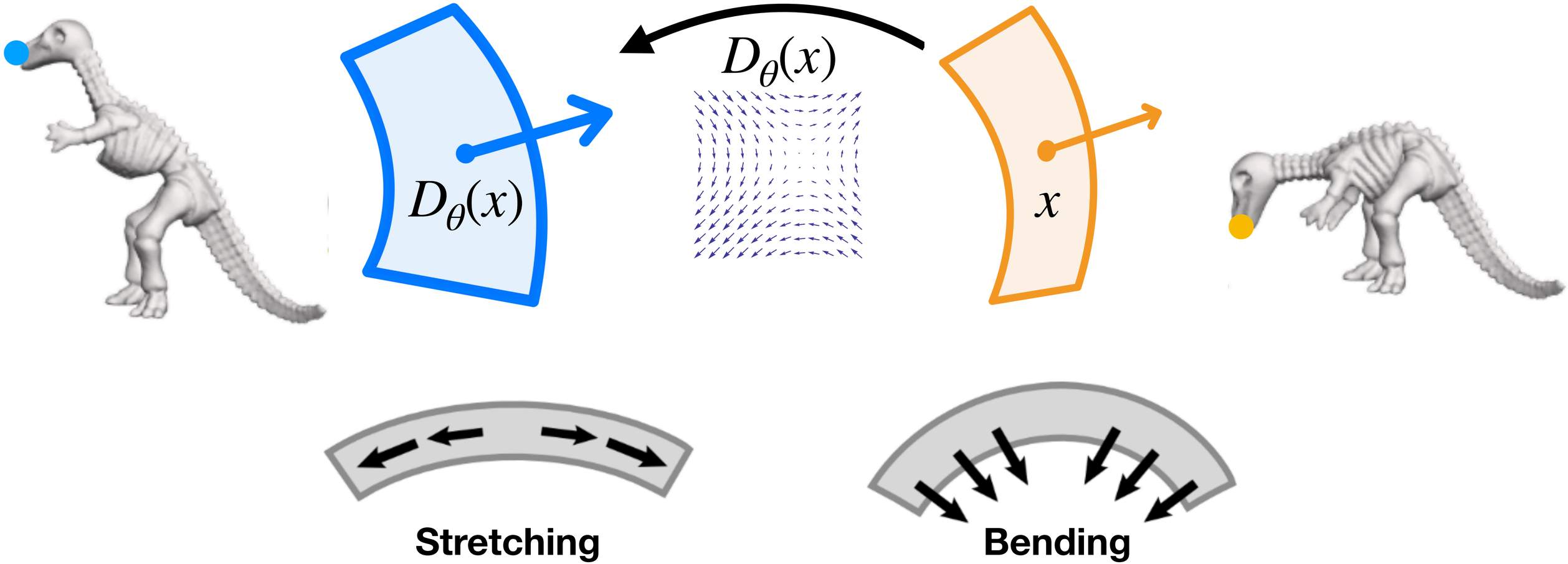


Positional Encoding

$G_\theta(x) = F(D_\theta(x))$

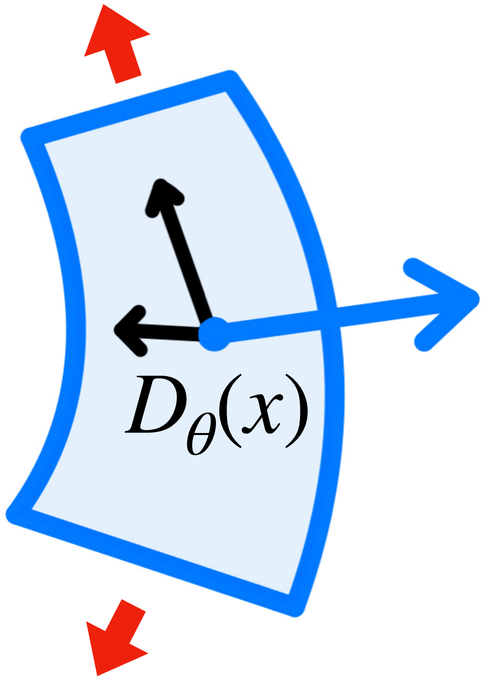


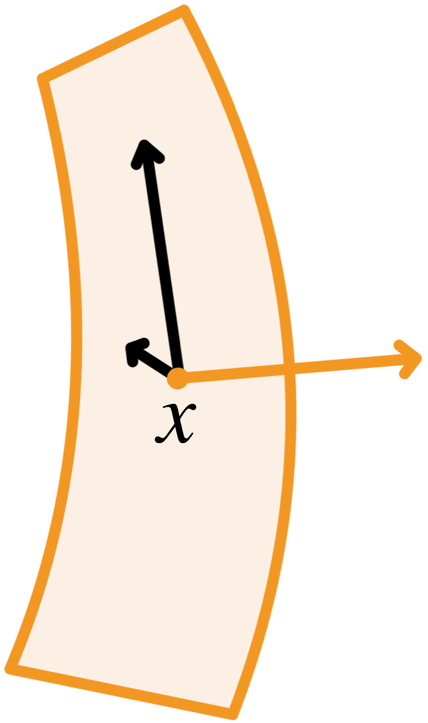
Step 3: Comparing



Step 3: Comparing

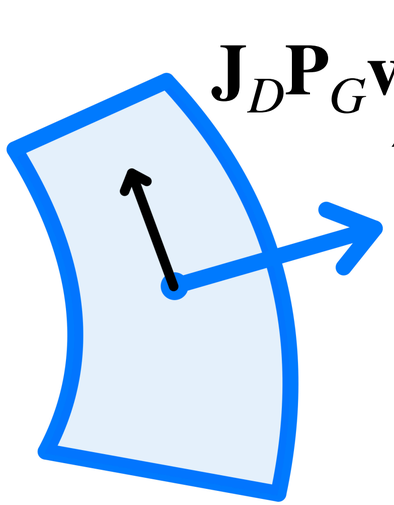
Stretch - change of tangent dot-product



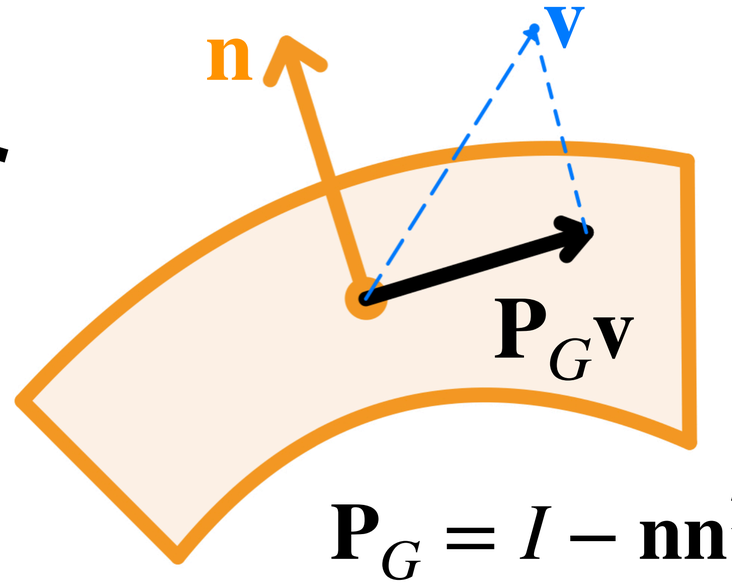


Step 3: Comparing

Stretch - change of tangent dot-product



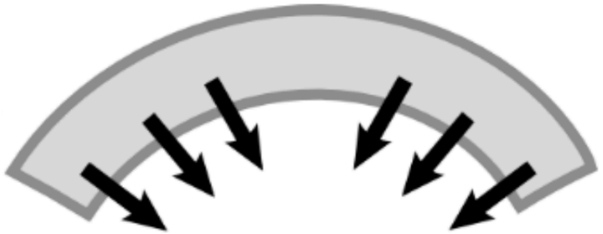
$D_\theta(x)$



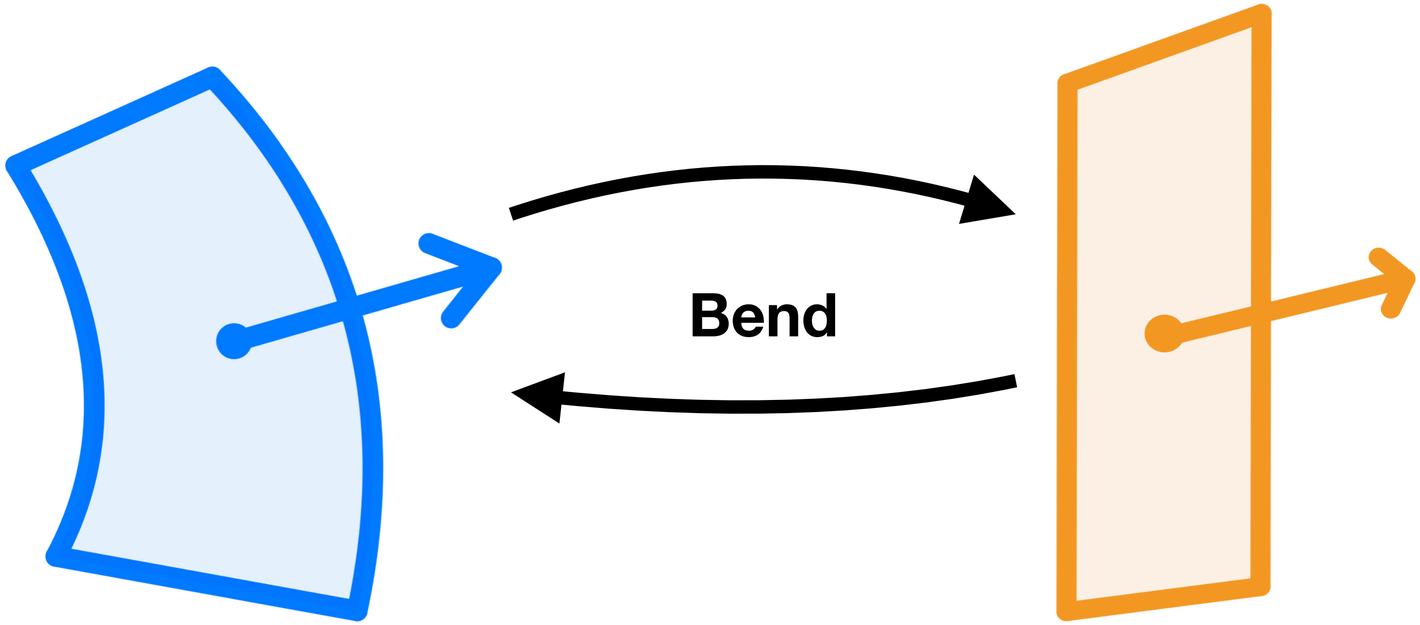
$$\mathbf{P}_G = I - \mathbf{n}\mathbf{n}^T$$

$$\mathscr{L}_s = \int \left\| \mathbf{P}_G^T \left(\mathbf{I} - \mathbf{J}_D^T \mathbf{J}_D \right) \mathbf{P}_G \right\|_F dx$$

Step 3: Comparing



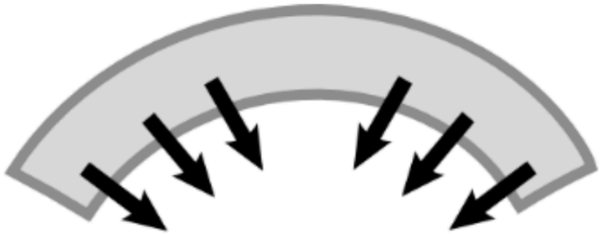
Bending - change of curvature



More curved

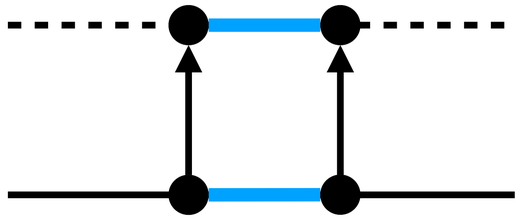
Less curved

Step 3: Comparing



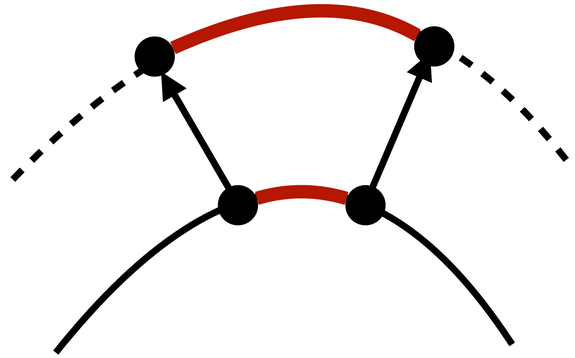
Bending - change of **curvature**

Curvature - change along normal direction



Low curvature

Little change

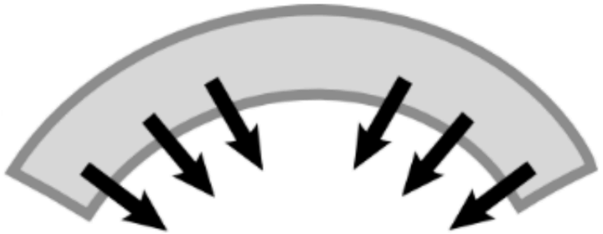


High curvature

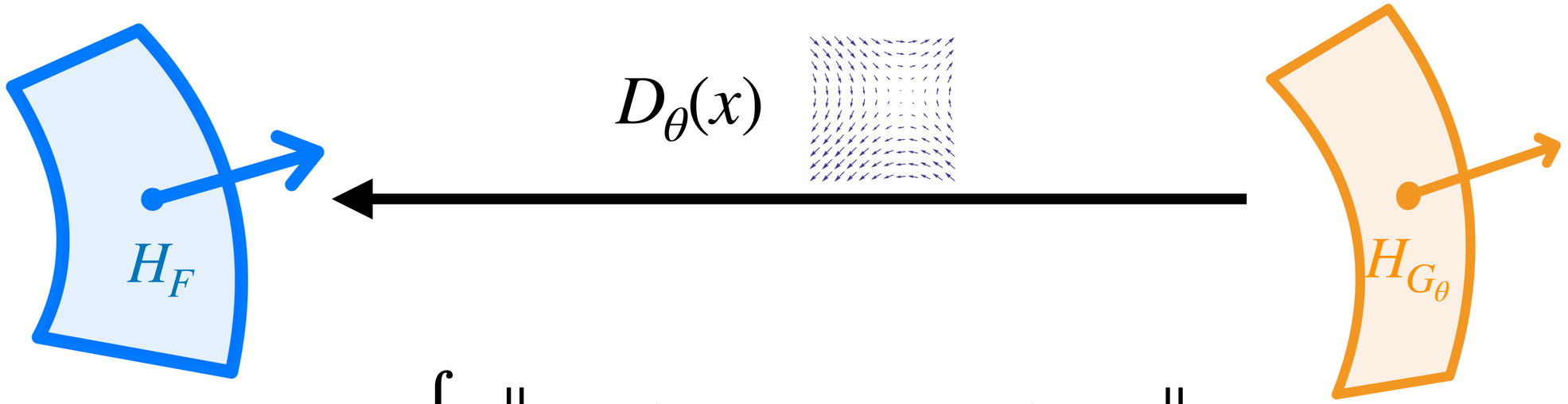
Large change

$$H_f = \begin{bmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{yx} & f_{yy} & f_{yz} \\ f_{zx} & f_{zy} & f_{zz} \end{bmatrix}$$

Step 3: Comparing



Bending - change of curvature



$$\mathscr{L}_b = \int_x \left\| \mathbf{P}_G^T \left(\mathbf{H}_G - \mathbf{J}_D^T \mathbf{H}_F \mathbf{J}_D \right) \mathbf{P}_G \right\|_F dx$$

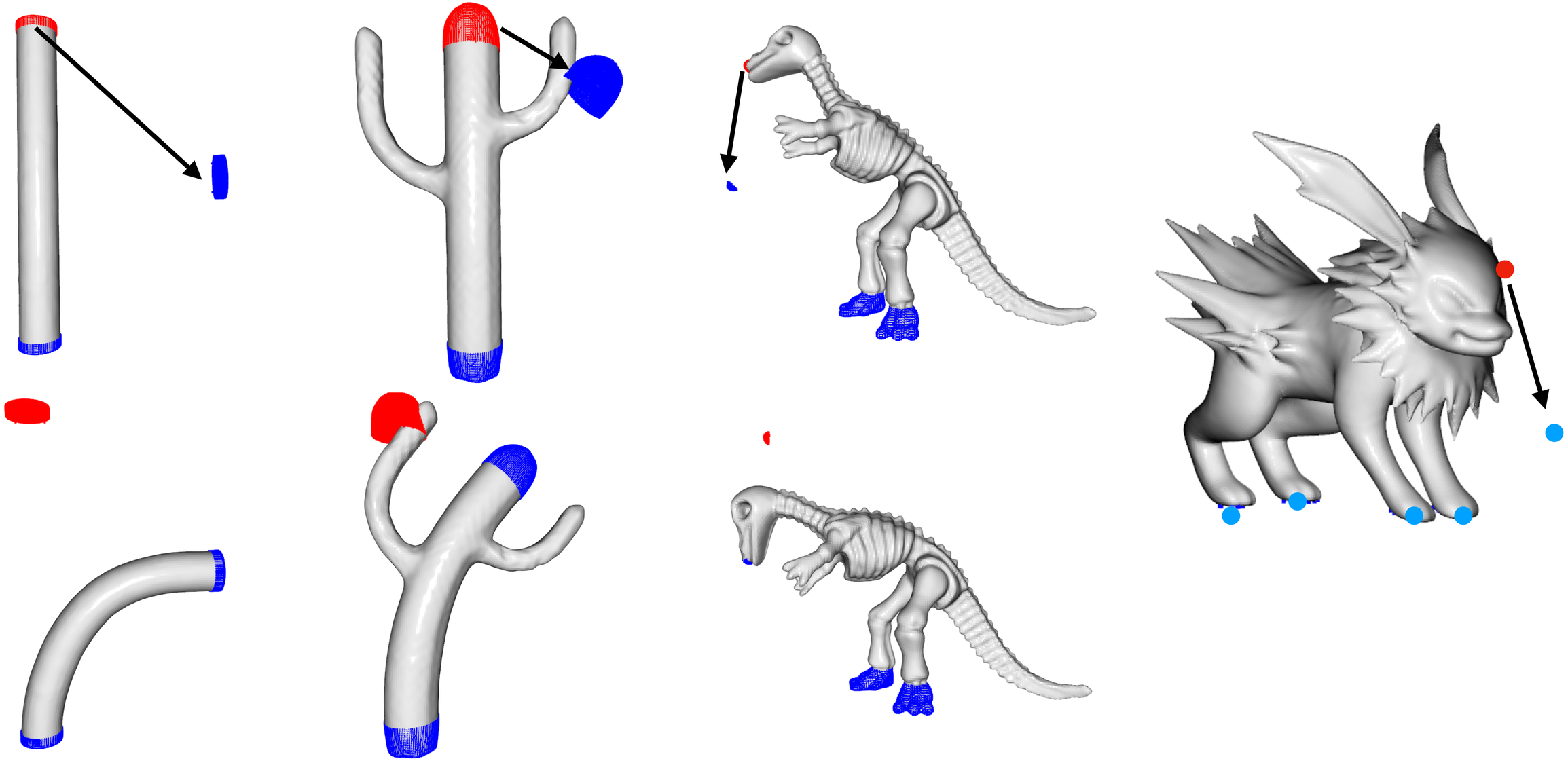
Step 3: Final Objective

$$\mathscr{L}_s = \int \left\| \mathbf{P}_G^T \left(\mathbf{I} - \mathbf{J}_D^T \mathbf{J}_D \right) \mathbf{P}_G \right\|_F dx$$

$$\mathscr{L}_b = \int_x \left\| \mathbf{P}_G^T \left(\mathbf{H}_G - \mathbf{J}_D^T \mathbf{H}_F \mathbf{J}_D \right) \mathbf{P}_G \right\|_F dx$$

$$\min_\theta \; k_s \mathscr{L}_s + k_b \mathscr{L}_b + k_c \sum_i \max(\tau, |D_\theta(t_i) - h_i|)$$

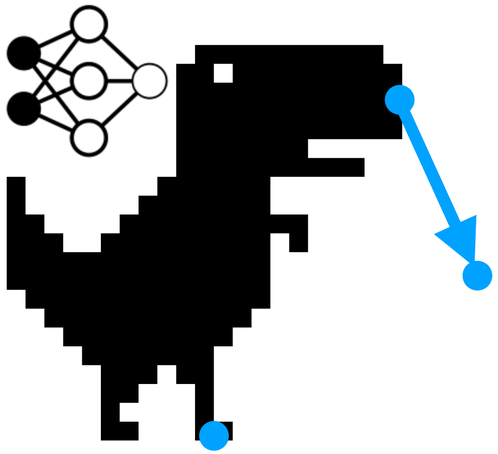
Deformation Results



Limitations & Future Works

- Speed
- Quality of the derivatives
- Field property preservation
- Physical Simulation

Speed

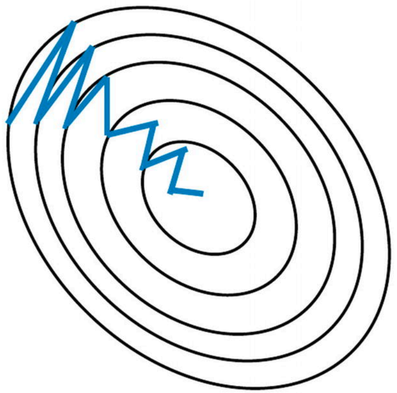


Input shape

User spec

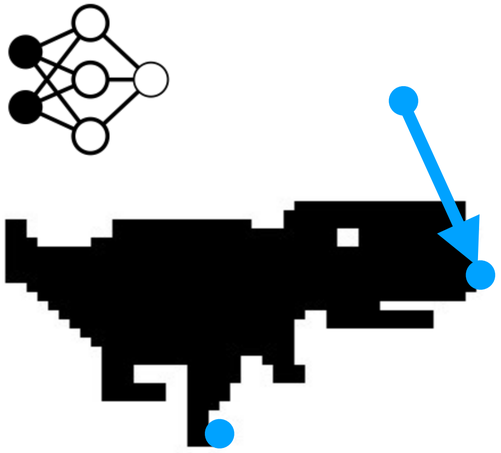
$$\min_{\theta} \mathscr{L}_{prior} + \mathscr{L}_{spec}$$

Optimization with SGD



Training time:

5 - 10 hours, 100k iterations

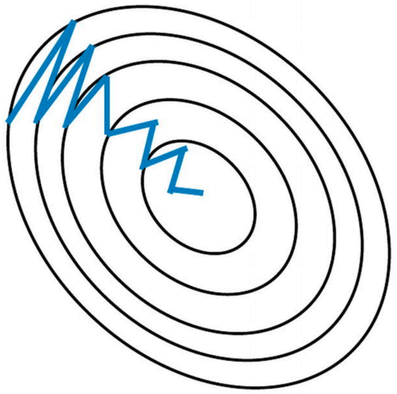


Output shape

Speed

$$\min_{\theta} \mathcal{L}_{prior} + \mathcal{L}_{spec}$$

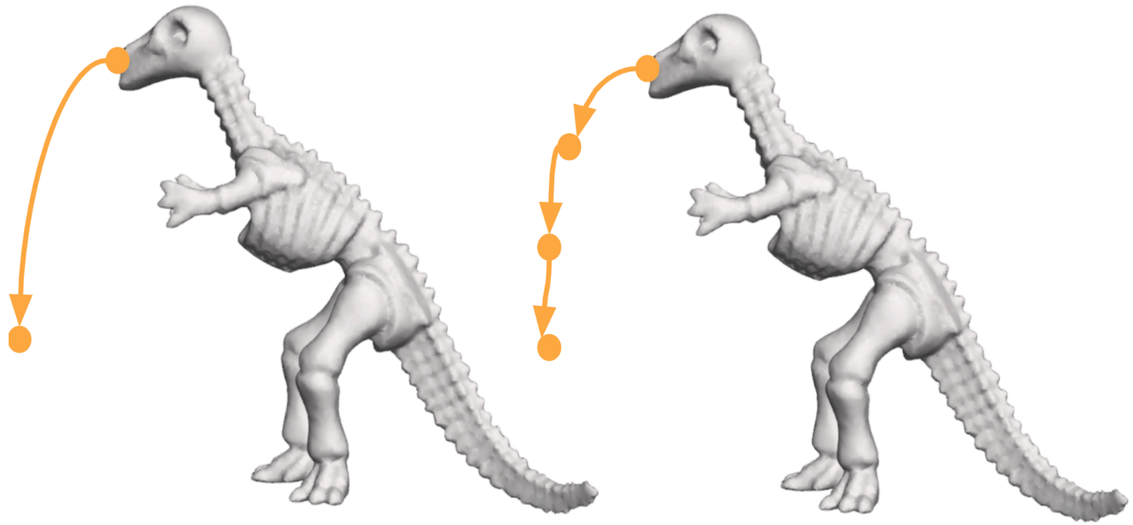
Optimization with SGD



Training time:

5 - 10 hours, 100k iterations

IDEA 1 : DIVIDE AND CONQUER

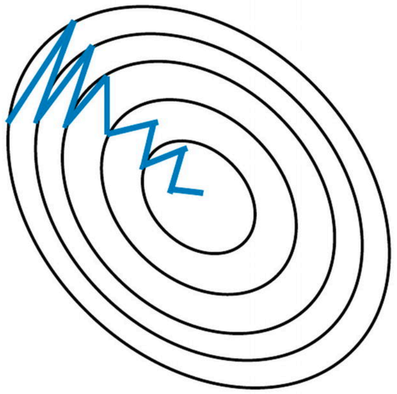


Observation: if the deformation is small, then it's easier to converge/ optimize

Speed

$$\min_{\theta} \mathcal{L}_{prior} + \mathcal{L}_{spec}$$

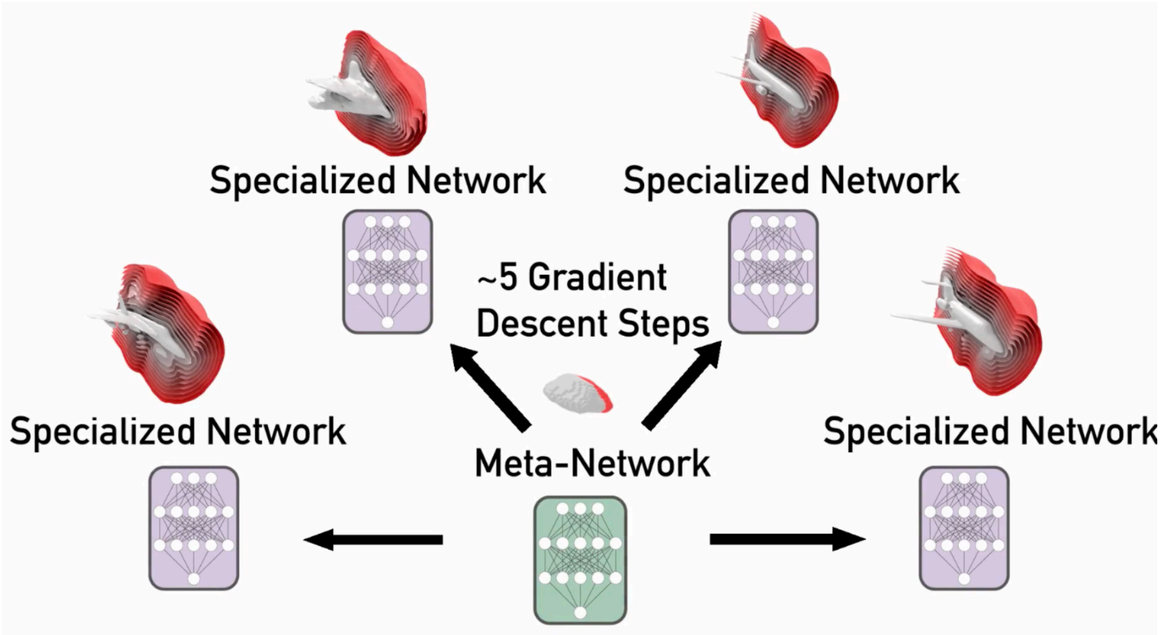
Optimization with SGD



Training time:

5 - 10 hours, 100k iterations

IDEA 2 : Don't start from scratch



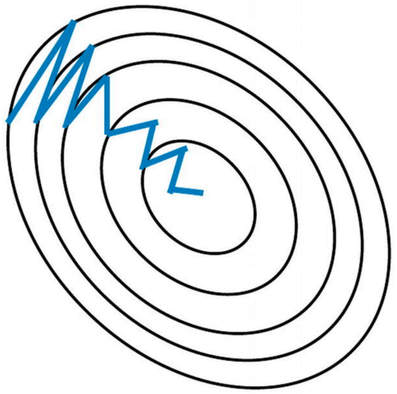
Observation: we can make a good guess of the output deformation field given the user edit

MetaSDF (Sitzmann et. al. 2021)

Speed

$$\min_{\theta} \mathcal{L}_{prior} + \mathcal{L}_{spec}$$

Optimization with SGD

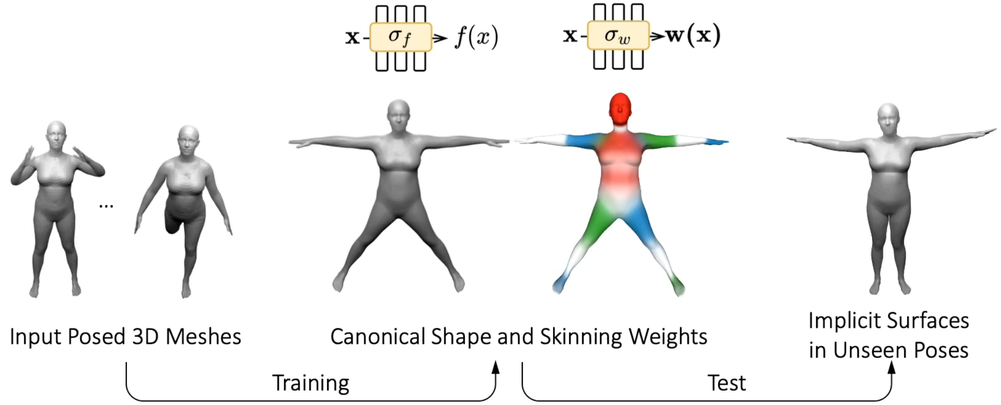


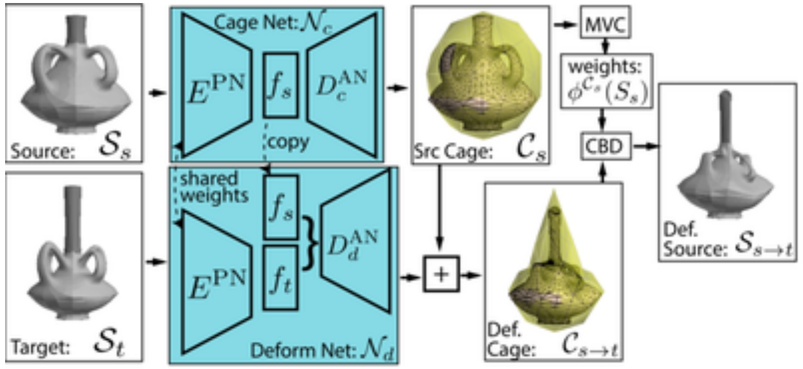
Training time:

5 - 10 hours, 100k iterations

IDEA 3: narrow down the deformation space

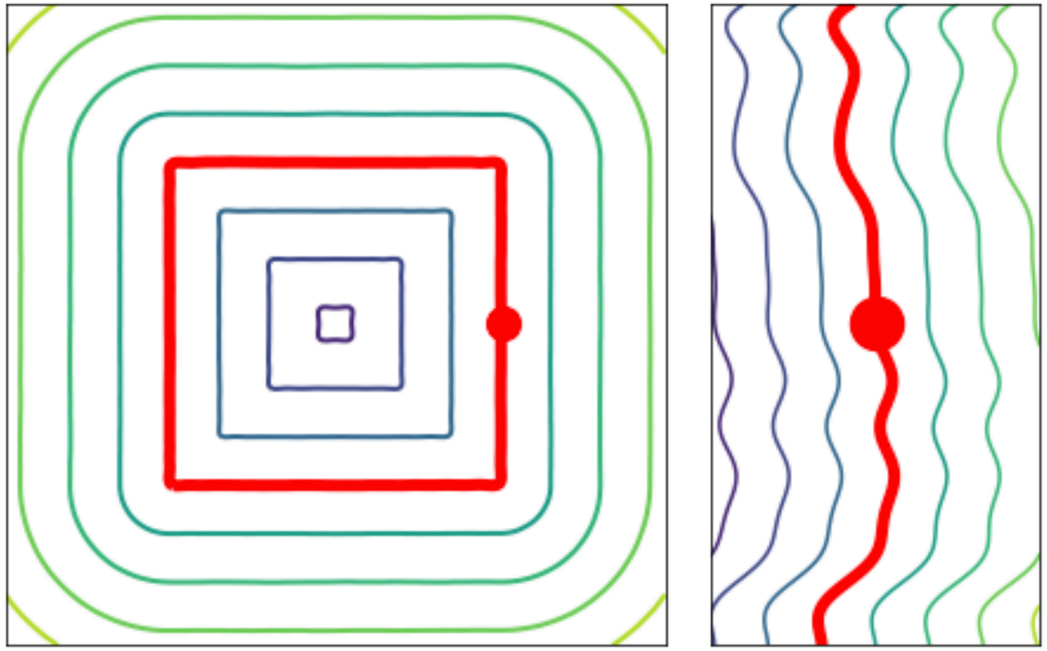
SNARF
(Chen et. al. 2021)



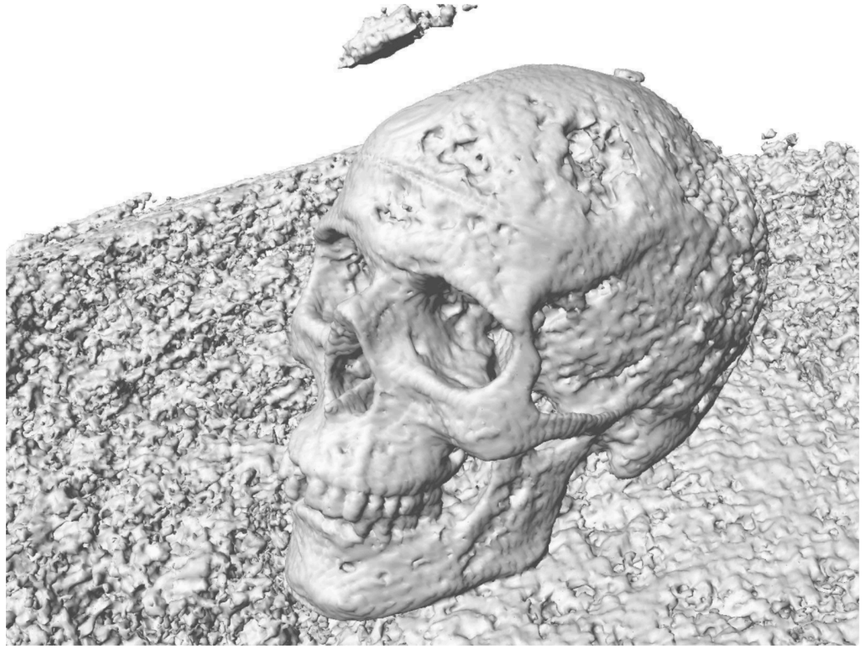


Neural Cage
(Wang et. al. 2020)

Quality of the Neural Fields Derivatives



L: SIREN learned to fit the SDF of a square; R: Zooming in on the surface.

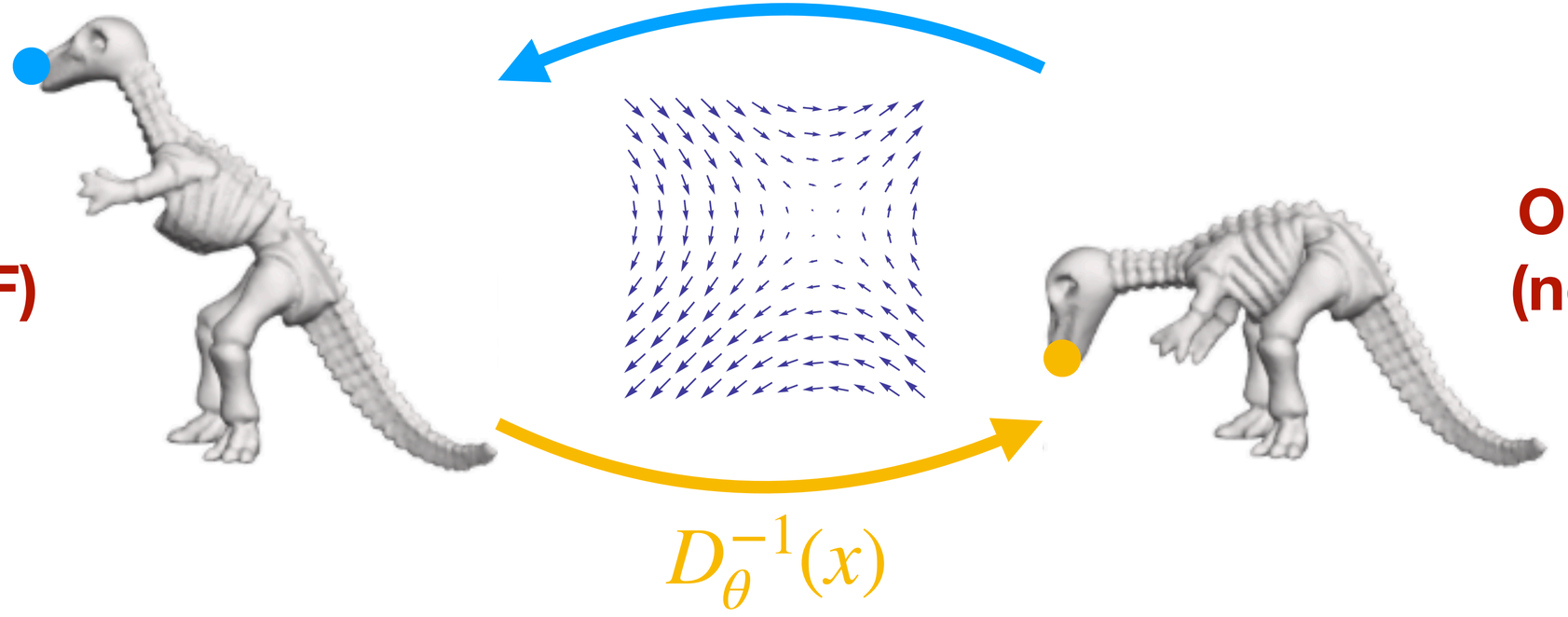


NeRF Surface at certain threshold (not tuned)

What kind of neural fields can provide good enough gradient information to be trained for geometry processing tasks?

Preservation of Field Properties

$F(x)$ $D_\theta(x)$ $G_\theta(x) = F(D_\theta(x))$



Input field (approx SDF)

Output field (not an SDF)

$D_\theta^{-1}(x)$

IDEA 1: Can we incorporate a “reinitialization” schema to output an SDF?

IDEA 2: We do need the field to be an SDF in order to perform the task?

Physics Simulation

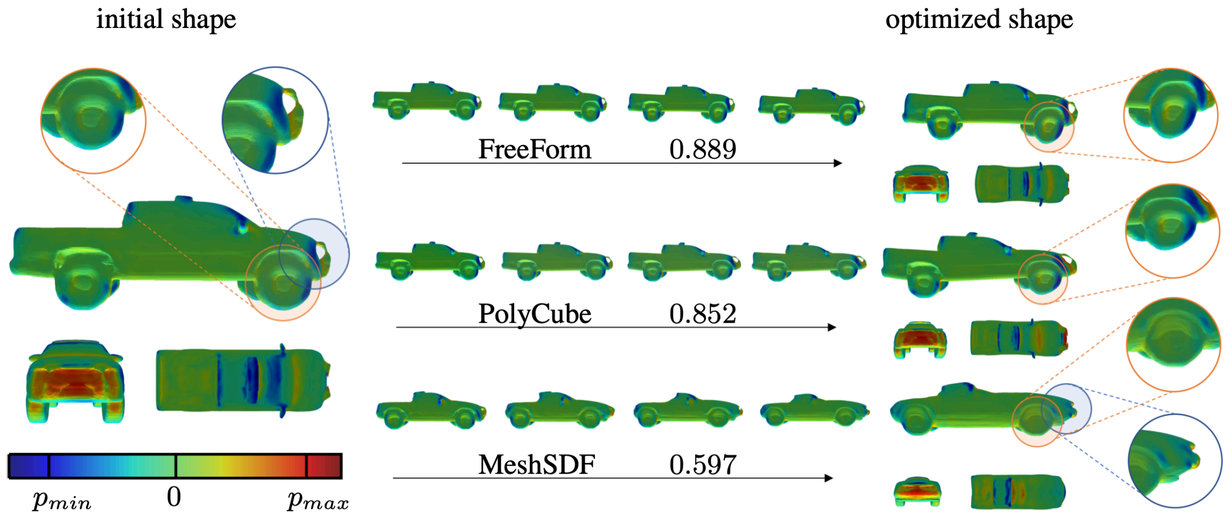


Figure 5: **Drag minimization.** Starting from an initial shape (left column), $\mathcal{L}_{task}$ is minimized using three different parameterizations: FreeForm (top), PolyCube (middle), and our *MeshSDF* (bottom). The middle column depicts the optimization process and the relative improvements in terms of $\mathcal{L}_{task}$. The final result is shown in the right column. FreeForm and PolyCube lack a semantic prior, resulting in implausible details such as sheared wheels (orange inset). By contrast, *MeshSDF* not only enforces such priors but can also effect topology changes (blue inset).

T: MeshSDF (Remelli et. al. 2021)

R: NeuralODE (Chen et. al. 2019)

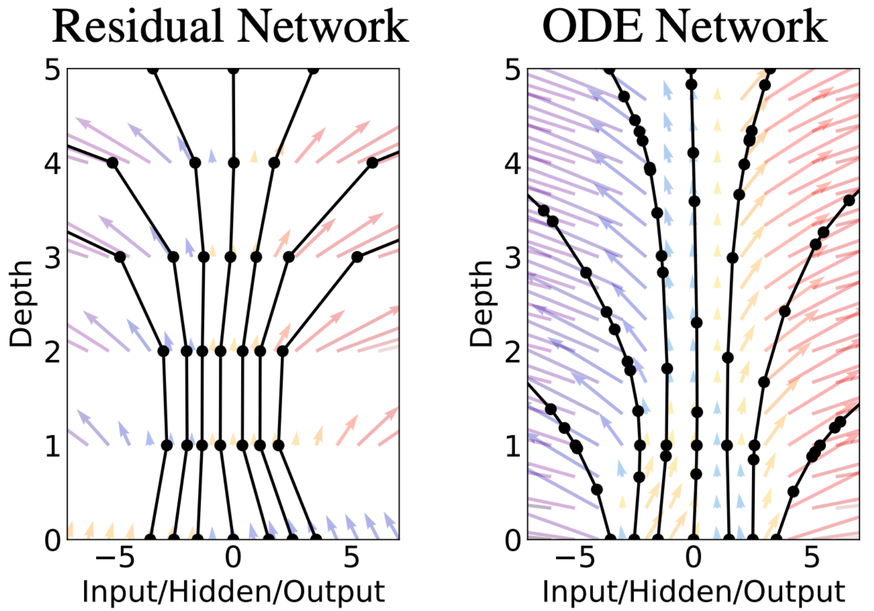


Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

Thanks for your time!

Project Page

https://www.guandaoyang.com/NFGP/

Code

https://github.com/stevenygd/NFGP

Discussion questions

- Any application scenario where using Neural Fields can be advantageous? (to represent the geometry, or potentially other physical quantities)
  - Need to represent a field in a compact way?
  - Need access to derivatives?
  - Can be benefited from data driven prior?
  - Might involve change of topology?

Discussion questions

- What do you see as a main obstacle to apply neural fields to problems you are interested in?
  - Optimization speed?
  - Numerical instability of the gradients?
  - Adding appearance/texture like mesh?