Implicit neural representations

Julie Digne

Master ID3D
LIRIS - CNRS
Équipe Origami

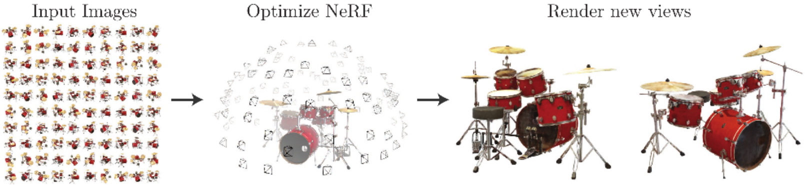
14/12/2023

The implicit alternative

- Instead of computing a triangulation, optimize an implicit field
- The implicit field is modeled by a neural network.

Outline

Neural Radiance Field (Nerf [Mildenhall et al. 2020]



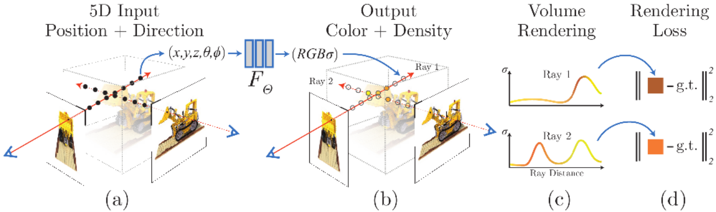
- Goal: Generate a new view from a set of views

Principle

Neural network takes as input a 3D coordinate and viewing direction and outputs the volume density and view-dependent emitted radiance at this location and direction.

- Cameras are calibrated (ie we know their positions, orientations and intrinsic parameters)

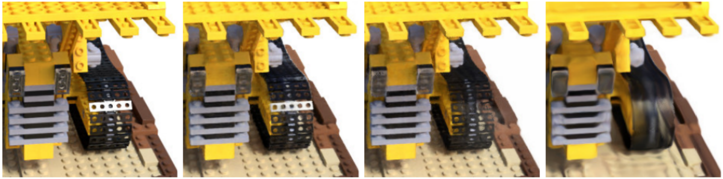
Training



- Neural net $F_\Theta : (x, y, z, \theta, \phi) \to (R, G, B, \sigma)$: Fully connected layers
- Volume rendering by querying along viewing directions.
- Sampling along the rays to estimate the rendering integral
- Comparison with the ground truth color on the target image

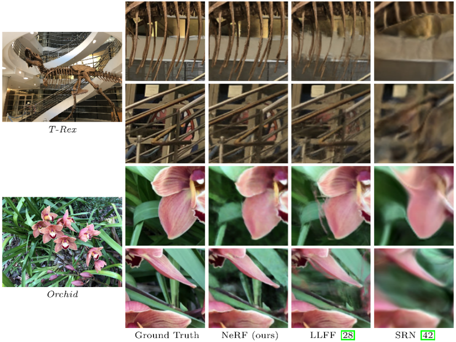
More tricks

- Add a positional encoding to improve high resolution details
- View-dependent radiance is what allows to capture mirror reflections



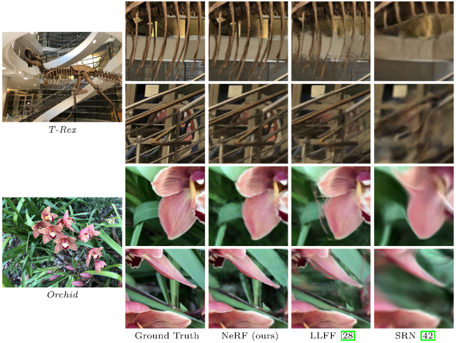
Ground Truth | Complete Model | No View Dependence | No Positional Encoding

Results



Video: `https://www.matthewtancik.com/nerf`

Results

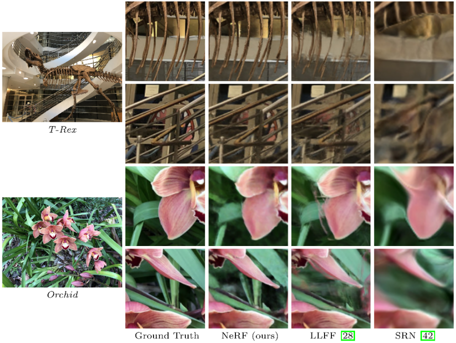


Video: `https://www.matthewtancik.com/nerf`

Training time

The optimization for a single scene typically take around 100– 300k iterations to converge on a single NVIDIA V100 GPU (about 1–2 days).

Results



Video: `https://www.matthewtancik.com/nerf`

Training time

The optimization for a single scene typically take around 100– 300k iterations to converge on a single NVIDIA V100 GPU (about 1–2 days). *(Faster variants released since: Instant NGP [Mueller 2022])*

Outline

Implicit neural field

- Model the signed distance field $u(x, y, z) = MLP_\theta(x, y, z)$ with θ the MLP parameters.

Implicit neural field

- Model the signed distance field $u(x, y, z) = MLP_\theta(x, y, z)$ with θ the MLP parameters.
- Signed distance field u to a surface S satisfies the Eikonal equation:

$$\|\nabla u\| = 1 \text{ with } u(x) = 0 \ \forall x \in \partial S$$

Implicit neural field

- Model the signed distance field $u(x, y, z) = MLP_\theta(x, y, z)$ with θ the MLP parameters.
- Signed distance field u to a surface S satisfies the Eikonal equation:

$$\|\nabla u\| = 1 \text{ with } u(x) = 0 \ \forall x \in \partial S$$

- Since a MLP is differentiable use the Eikonal equation as a loss function [Gropp 2020]

Optimization Process

- Input data a set of points $(x_i, \mathrm{n}_i), i \in I$
- Look for u continuous and a.e. $\mathcal{C}^1$ such that:

$$\begin{cases} \|\nabla u\| & = 1 \\ u_{|\partial\Omega} & = 0 \\ \nabla u_{|\partial\Omega]} & = \mathrm{n} \end{cases} \tag{1}$$

- Loss [Gropp 2020]

$$l(\theta) = \frac{1}{|I|} \sum_{i \in I} (|u_\theta(x_i)| + \tau \|\nabla u_\theta(x_i) - \mathrm{n}_i\|) + \lambda \mathbb{E}_x[(\|\nabla u_\theta(x)\| - 1)^2]$$

Periodic Activation Functions [Sitzmann 2021]

- Replace ReLU by periodic activation function $x \to \sin(\omega x)$. Better differentiability
- Loss:

$$\mathcal{L}_{sdf} = \frac{1}{|I|} \sum_{i \in I} (|u_\theta(x_i)| + \tau \|\nabla u_\theta(x_i) - \mathrm{n}_i\|) + \lambda \mathbb{E}_x[(\|\nabla u_\theta(x)\| - 1)^2] + \lambda_2 \mathbb{E}_{x \notin \Omega}[(\|\psi(u_\theta(x)\|]$$

with $\psi(u_\theta(x)) = \exp -\alpha|u_\theta(x)|;\ \alpha >> 1$

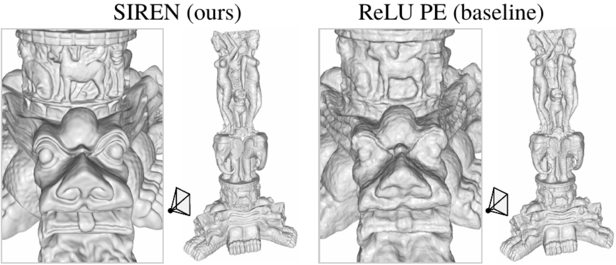


Figure 4: A comparison of SIREN used to fit a SDF from an oriented point clouse against the same fitting performed by an MLP using a ReLU PE (proposed in [35]).

From [Sitzmann 2020]

Periodic Activation Functions [Sitzmann 2021]

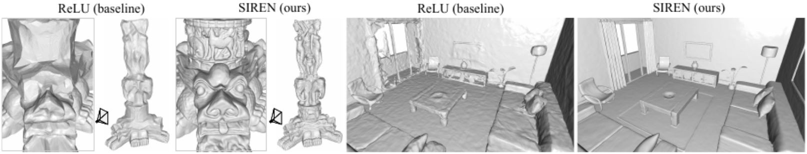
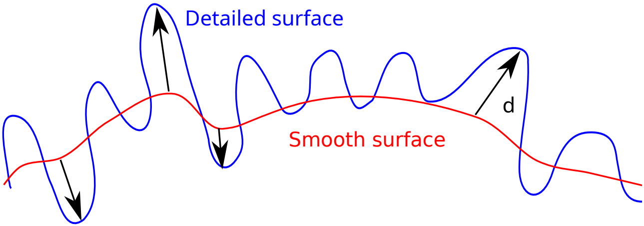


Figure 4: Shape representation. We fit signed distance functions parameterized by implicit neural representations directly on point clouds. Compared to ReLU implicit representations, our periodic activations significantly improve detail of objects (left) and complexity of entire scenes (right).

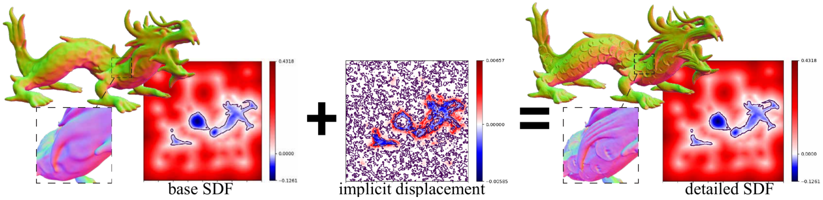
From [Sitzmann 2020]

Implicit displacement field [Yifan 2022]



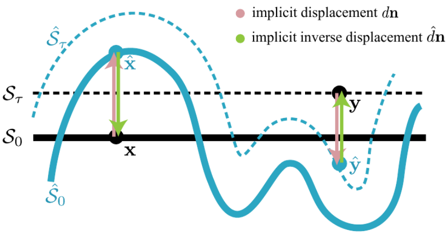
- Decompose the surface into a smooth base and a displacement field
- *Both* the smooth surface and the displacement field are learned

Overview



[Yifan 2022]

Implicit displacement field - definition



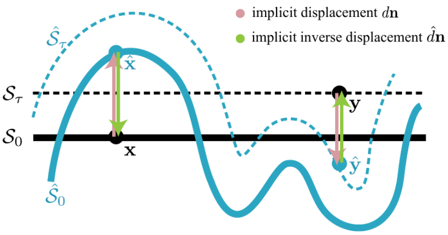
[Yifan 2022]

Definition

Smooth base SDF f, detailed SDF $\hat{f}$, an implicit displacement field (IDF)

$$f(x) = \hat{f}(x + d(x)n), \text{ where } n = \frac{\nabla f(x)}{\|\nabla f(x)\|}$$

Implicit displacement field - definition



[Yifan 2022]

Definition

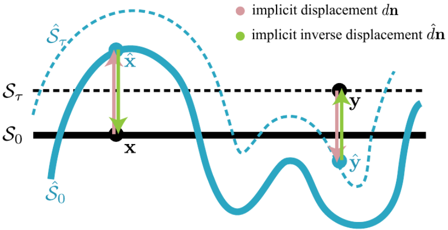
Smooth base SDF f, detailed SDF $\hat{f}$, an implicit displacement field (IDF)

$$f(x) = \hat{f}(x + d(x)n), \text{ where } n = \frac{\nabla f(x)}{\|\nabla f(x)\|}$$

Learning - naive version

Minimize at query points $x \in \mathbb{R}^3$: $|f(x) - f_{GT}(\hat{x})|$ with $\hat{x} = x + d(x)n$

Inverse implicit displacement field

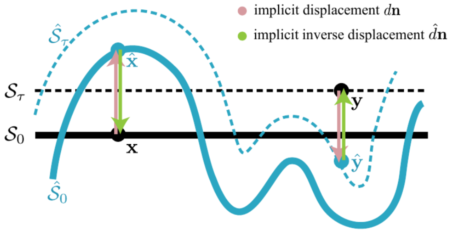


[Yifan 2022]

Alternative

Inverse Displacement Mapping $\hat{d}$: $f(x + \hat{d}(\hat{x})n) = \hat{f}(\hat{x})$

Inverse implicit displacement field



[Yifan 2022]

Alternative

Inverse Displacement Mapping $\hat{d}$: $f(x + \hat{d}(\hat{x})n) = \hat{f}(\hat{x})$

- One can use $\hat{n} = \frac{\nabla f(\hat{x})}{\|\nabla f(\hat{x})\|}$ instead of $\hat{n} = \frac{\nabla f(\hat{x})}{\|\nabla f(\hat{x})\|}$ (error is theoretically bounded)

Architecture and training

- Two SIREN networks, with different ω parameters (one low - base, one high - idf)

Composed distance field

$$f(x) = \mathcal{N}_{\omega_B}(x)$$

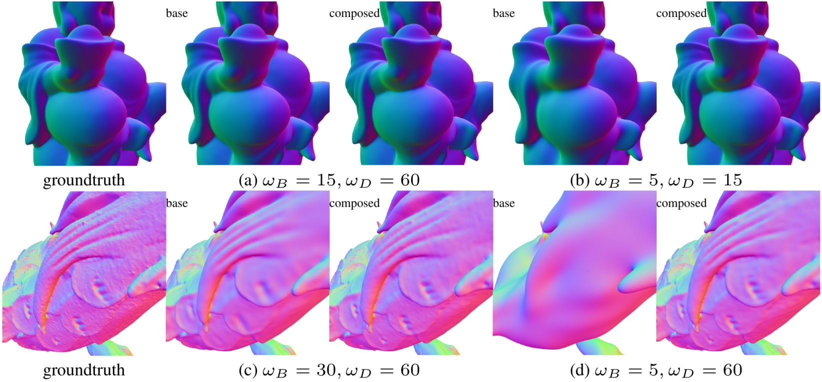
$$\hat{f}(x) = \mathcal{N}_{\omega_B}(x + \chi(f(x))\mathcal{N}_{\omega_D}(x)\frac{\nabla f(x)}{\|\nabla f(x)\|})$$

where χ is an attenuation function

Loss

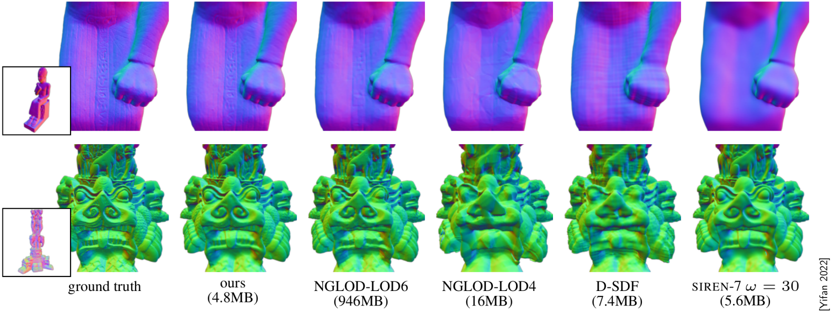
$$\mathcal{L}_{\hat{f}} = \sum_{x \in \mathbb{R}^3} \lambda_0 |\|\nabla \hat{f}(x)\| - 1| + \sum_{(p,n) \in \partial\Omega} (\lambda_1 |\hat{f}(p)| + \lambda_2 (1 - \langle \nabla \hat{f}(p), n \rangle)) + \sum_{x \in \mathbb{R}^3} \lambda_3 \exp(-100 \hat{f}(x))$$

Results - Surface decomposition

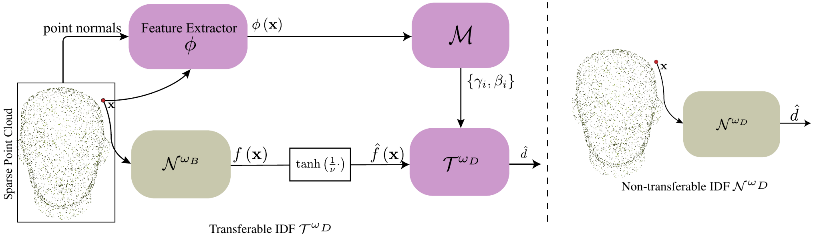


[Yifan 2022]

Detailed surface reconstruction

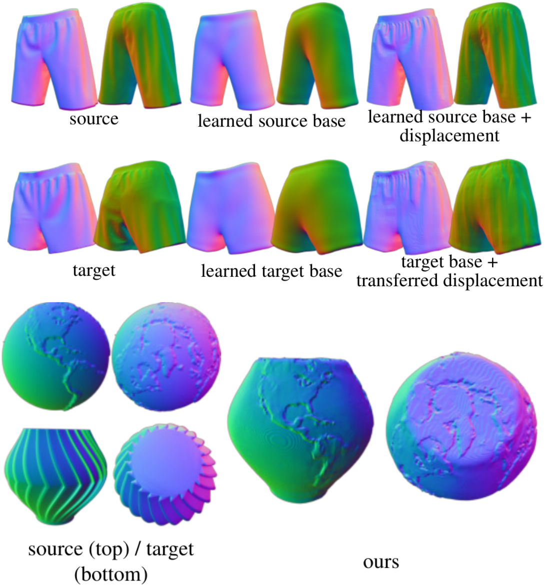


Detail transfer



[Yifan 2022]

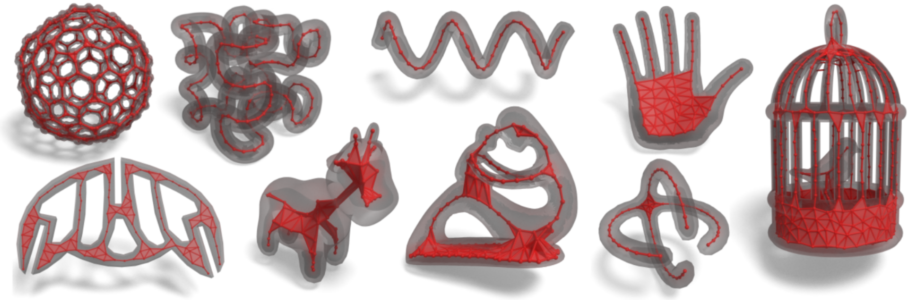
Detail transfer results



[Yifan 2022]

Outline

Regularizing INR away from the surface

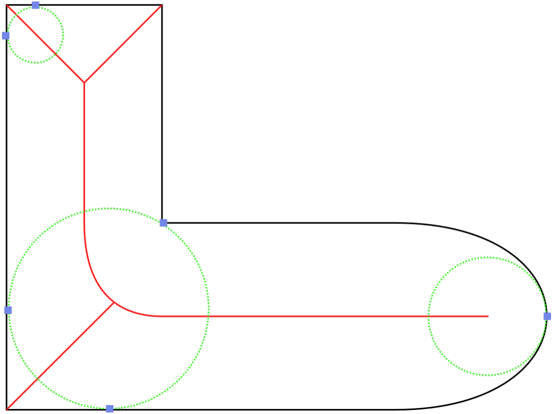


[Clémot, Digne 2023]

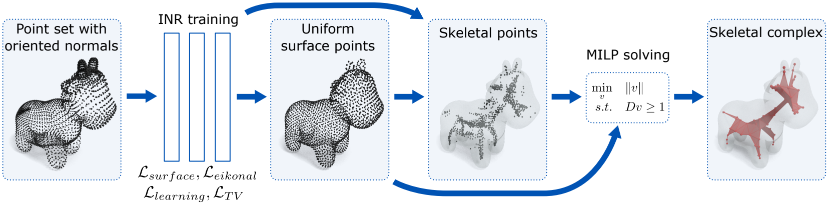
Medial Axis

Definition

A point p belongs to the medial axis of a compact shape if it has at least two distinct nearest neighbors on the shape surface.



Overview



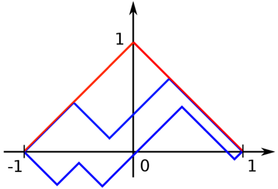
Point set with oriented normals
INR training
$\mathcal{L}_{surface}, \mathcal{L}_{eikonal}$
$\mathcal{L}_{learning}, \mathcal{L}_{TV}$
Uniform surface points
Skeletal points
MILP solving
$\min_v \quad \|v\|$
$s.t. \quad Dv \geq 1$
Skeletal complex

Eikonal Equation

- Infinite number of solutions
- Viscosity solution theory: allows to select the right solution
- Use smooth eikonal equation (not practical [Lipman 2019])

$$\|\nabla u\| - \varepsilon \Delta u = 1$$

- Consequence: blobs appear



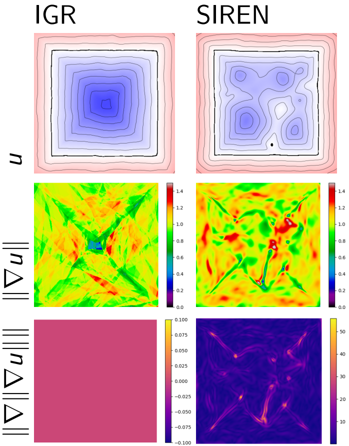
[Camilli 2014]

Infinite nber of solutions

Not an issue close to the surface – but far away?

Which neural network?

- MLP (6 layers, 128-256 neurons/layer) with ReLU activation functions
- ReLU yields a function in $W^{1,p}$ [Lipman 2019]
- But: not always easy to train
- Sitzman (2021) replaces ReLU with sine activation function: smooth function



TV regularization - some theory

- Look for a smooth surrogate for the signed distance function
- Medial axis: zeros of the gradient
- The TV term favors that u has no second order differential content along the gradient lines

Since $\nabla u = (u_x, u_y, u_z)$, it follows:

$$\begin{aligned}
\nabla \|\nabla u\| &= \nabla \sqrt{u_x^2 + u_y^2 + u_z^2} \\
&= \frac{1}{2\|\nabla u\|} \begin{pmatrix} 2u_x u_{xx} + 2u_y u_{xy} + 2u_z u_{xz} \\ 2u_x u_{xy} + 2u_y u_{yy} + 2u_z u_{yz} \\ 2u_x u_{zx} + 2u_y u_{zy} + 2u_z u_{zz} \end{pmatrix} \\
&= H_u \frac{\nabla u}{\|\nabla u\|}
\end{aligned}$$

Total loss

- Eikonal loss:

$$\mathcal{L}_{eikonal} = \int_{\mathbb{R}^3} (1 - \|\nabla u(p)\|)^2 \, dp \qquad (2)$$

- Surface loss:

$$\mathcal{L}_{\text{surface}} = \int_{\partial\Omega} u(p)^2 dp + \int_{\partial\Omega} 1 - \frac{\mathrm{n}(p) \cdot \nabla u(p)}{\|\mathrm{n}(p)\| \, \|\nabla u(p)\|} dp \qquad (3)$$

- Learning point loss

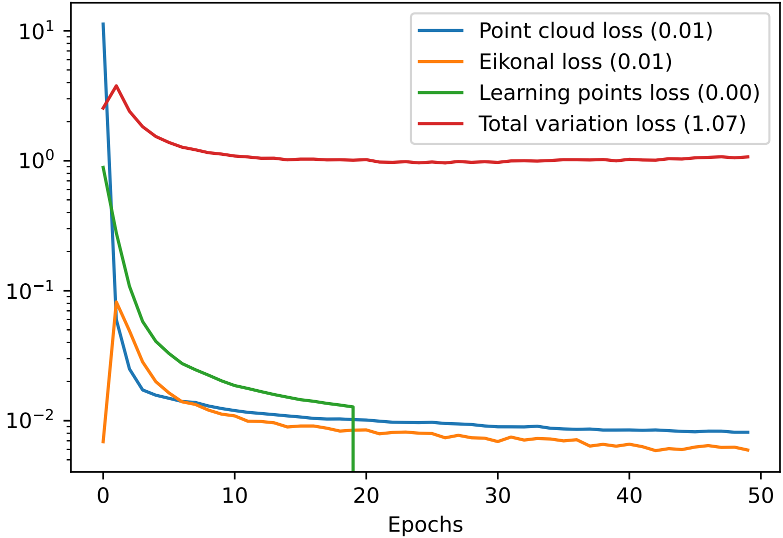
$$\mathcal{L}_{\text{learning}} = \sum_{p \in \mathcal{P}} (u(p) - d(p))^2 + \sum_{p \in \mathcal{P}} 1 - \frac{\nabla u(p) \cdot \nabla d(p)}{\|\nabla u(p)\| \, \|\nabla d(p)\|} \qquad (4)$$

- + TV loss

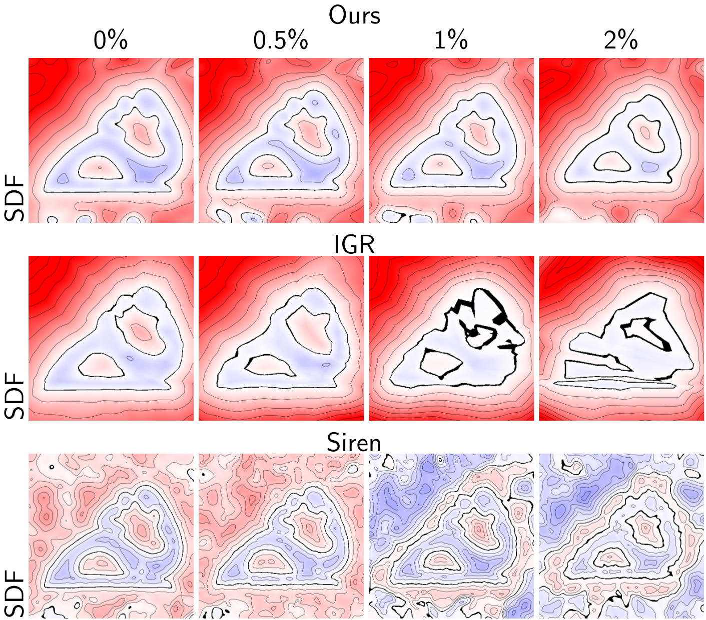
Loss

$$\mathcal{L} = \lambda_e \mathcal{L}_{eikonal} + \lambda_s \mathcal{L}_{surface} + \lambda_l \mathcal{L}_{learning} + \lambda_{TV} \mathcal{L}_{TV} \qquad (5)$$

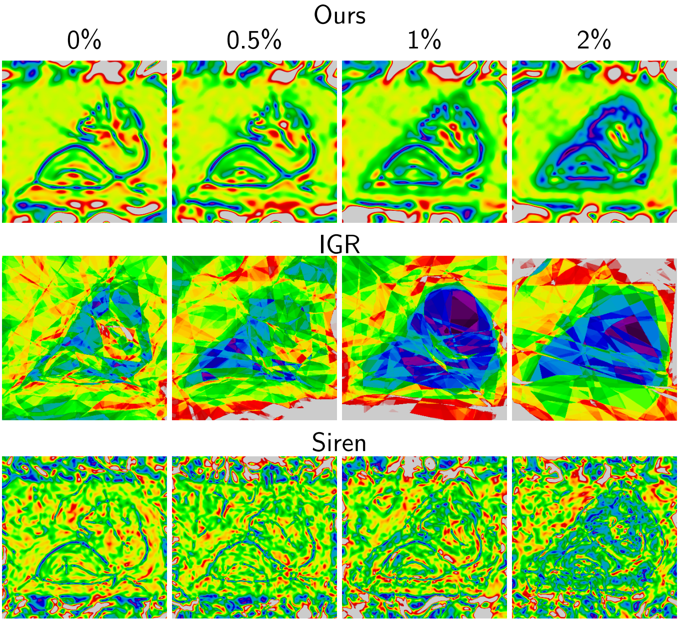
Convergence



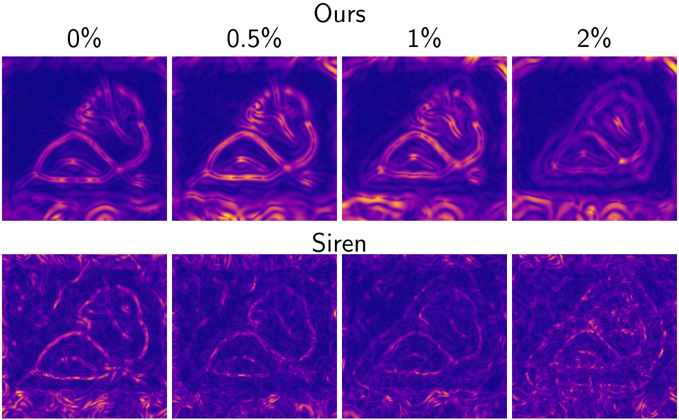
Resulting Fields



$\|\nabla u\|$



$\nabla \|\nabla u\|$



then...

- GPU skeleton tracing to extract points on the skeleton
- Select a subset based on the Coverage Axis method [Dou 2022]
  - N points x_i, M skeletal points s_i with distance r_i to the surface.
  - Coverage matrix: D $(N \times M)$

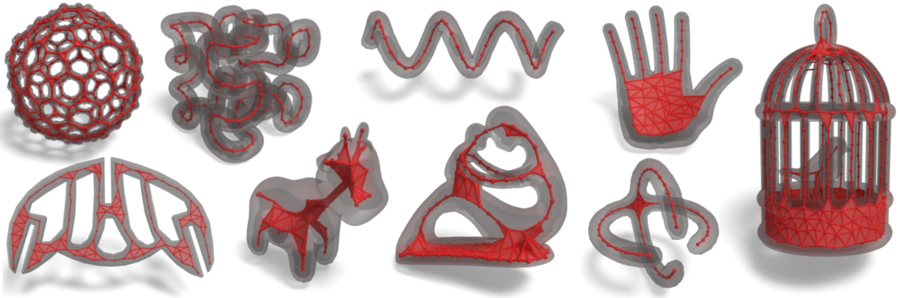
$$D_{ij} = 1 \text{ if } \|p_i - s_j\| - r_j \leq \delta \text{ and } 0 \text{ otherwise}$$

  - Mixed Integer Linear Problem:

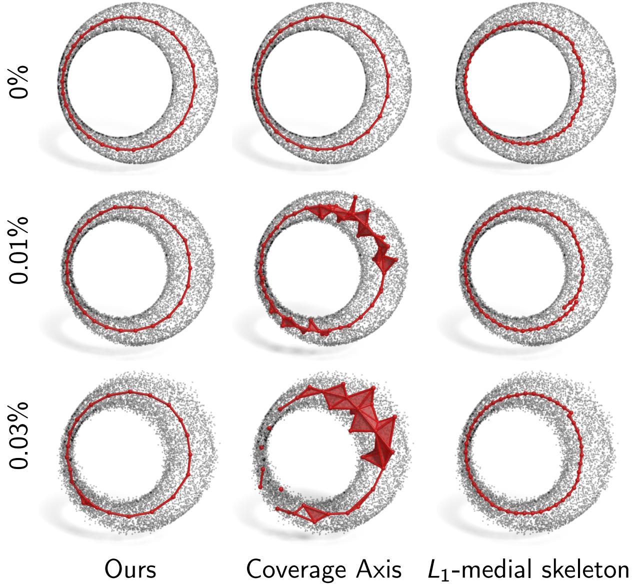
$$\begin{array}{rl} \min & \|v\|_2 \\ \text{s.t.} & Dv \succeq 1 \end{array} \tag{6}$$

- Link the selected points by computing the regular triangulation of weighted skeletal points and surface points + keep simplices between skeletal points

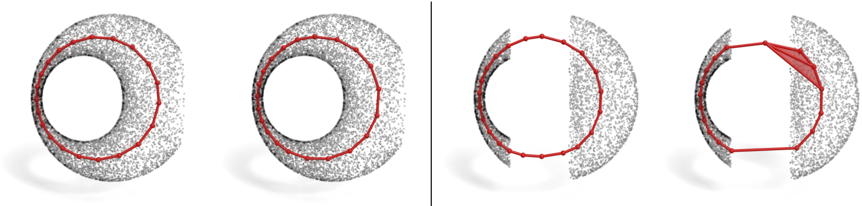
Results



Results

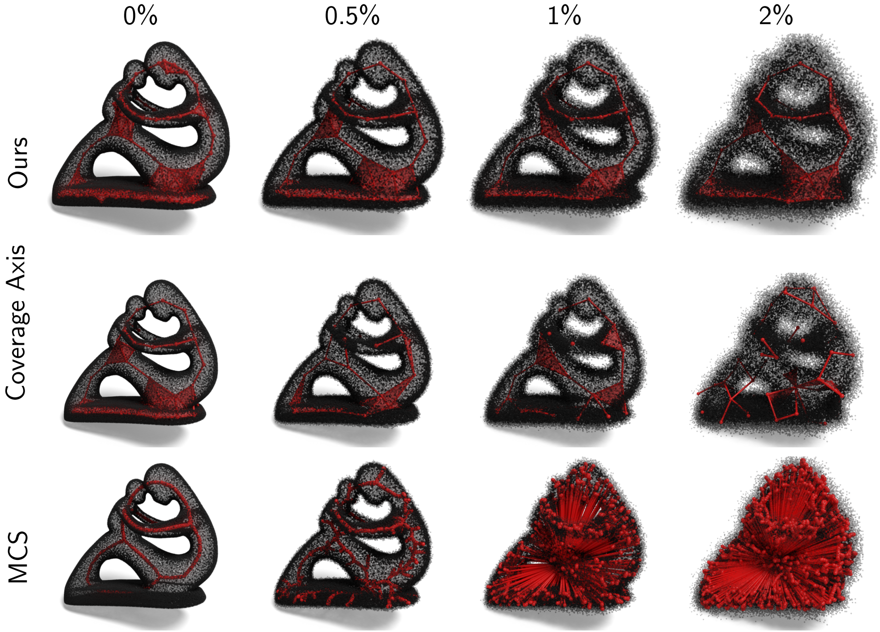


Results

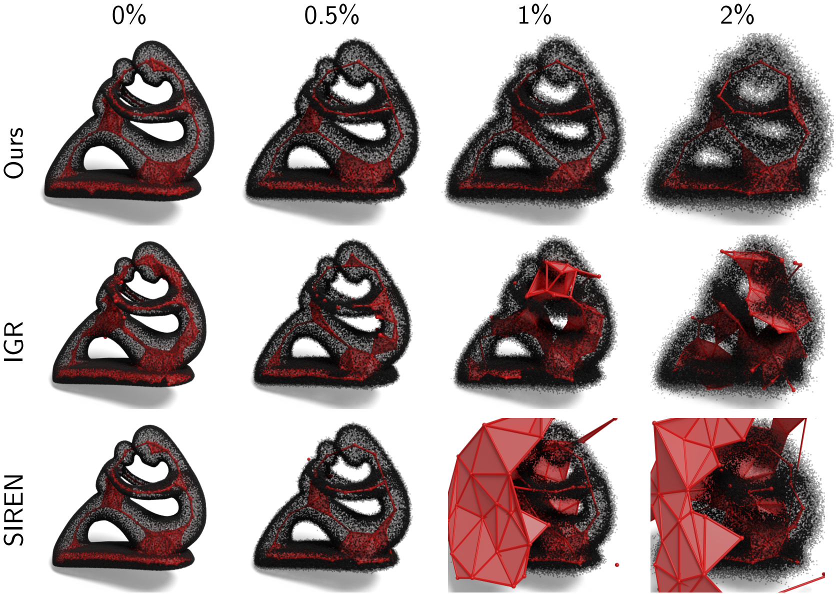


Ours | Coverage Axis | Ours | Coverage Axis

With noise



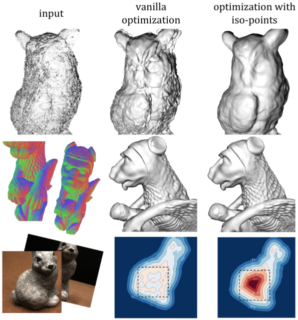
With noise



Outline

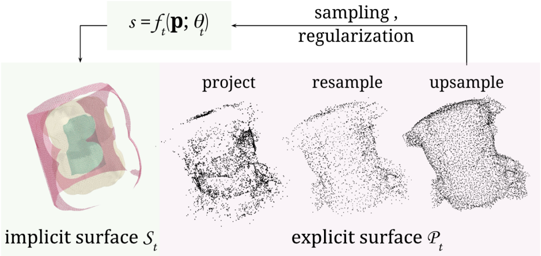
Projecting points on the surface [Yifan 2021]

- Sample points on a neural implicit
- Use them to improve robustness and accuracy



[Yifan 2021]

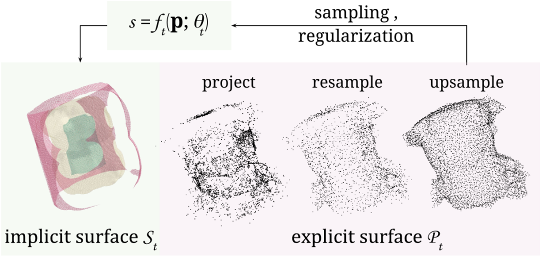
Projection on the surface



[Yifan 2021]

- Starting from a point q_0 in $\mathbb{R}^3$ project it on the surface
- Newton Iterations: $q_{k+1} = q_k - J_f^+(q_k) f_\theta(q_k)$ with $J_f^+(q_k) = \frac{1}{\|J_f(q_k)\|^2} J_f(q_k)$
- For nonsmooth fields, set an upper threshold for the displacement magnitude

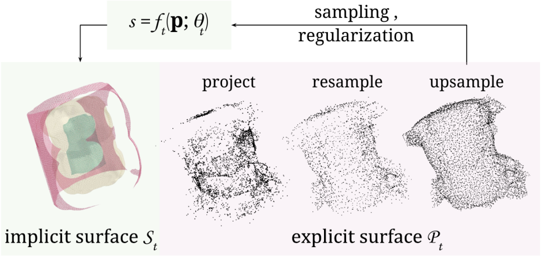
Uniform resampling



[Yifan 2021]

- Move the points away from dense areas $\tilde{q} \leftarrow \tilde{q} - \alpha r$
- α step size
- $r = \sum_{\tilde{q}_i \in \mathcal{N}(\tilde{q})} w(\tilde{q}_i, \tilde{q}) \frac{\tilde{q}_i - \tilde{q}}{||\tilde{q}_i - \tilde{q}||}$

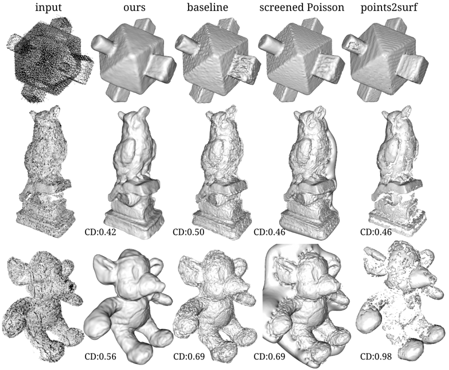
Upsampling



[Yifan 2021]

- Move the points away from the edges (Edge-away resampling [Huang 2011])
- Each point is :
  - attracted to points that have a similar normal
  - repulsed from dense areas.
- Upsampled points are reprojected on the surface

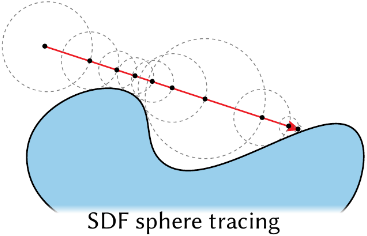
Application to INR fitting regularization

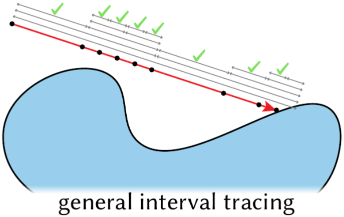


[Yifan 2021]

- Warmup training (300 iterations)
- Extract isopoints + add isopoints to data points
- Update the isopoints every 1000 iterations

Arithmetic Queries [Sharp 2022]

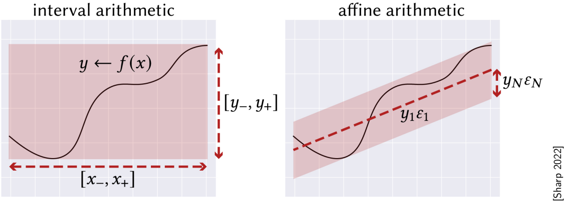




[Sharp 2022]

- f_θ a neural implicit *Not necessarily a signed distance field*.
- Sphere tracing for SDF, interval arithmetic for general implicit field.
- Goal: adapt interval arithmetic for neural implicits.

Affine arithmetic [Comba and Stolfi 1993]



- Interval arithmetic gives loose bounds
- Affine arithmetic: tracks affine coefficients through computation
- Similar to forward auto-diff: linear operations, nonlinear operations by linearization (adds affine coefficients!)

MLP

Affine operations followed by ReLU nonlinearity

Nonlinearities

- $\hat{x} = x_0 + \sum_{i=1}^{N} x_i \varepsilon_i \;\; \varepsilon_i \in [-1, 1]$
- Replace f by a linear approximation $\hat{f}(x) \approx \alpha x + \beta$
- $\gamma = \max_{x \in range(\hat{x})} |f(x) - \hat{f}(x)|$

Nonlinearities

- $\hat{x} = x_0 + \sum_{i=1}^{N} x_i \varepsilon_i \ \ \varepsilon_i \in [-1, 1]$
- Replace f by a linear approximation $\hat{f}(x) \approx \alpha x + \beta$
- $\gamma = \max_{x \in range(\hat{x})} |f(x) - \hat{f}(x)|$
- <2>$\hat{y} = f(\hat{x}) = \alpha x_0 + \beta + \sum_{i=1}^{N} \alpha x_i \varepsilon_i + \gamma \varepsilon_{N+1}$

Nonlinearities

- $\hat{x} = x_0 + \sum_{i=1}^{N} x_i \varepsilon_i \;\; \varepsilon_i \in [-1, 1]$
- Replace f by a linear approximation $\hat{f}(x) \approx \alpha x + \beta$
- $\gamma = \max_{x \in range(\hat{x})} |f(x) - \hat{f}(x)|$
- <2>$\hat{y} = f(\hat{x}) = \alpha x_0 + \beta + \sum_{i=1}^{N} \alpha x_i \varepsilon_i + \gamma \varepsilon_{N+1}$
- Each layer with width W adds W new coefficients.

Nonlinearities

- $\hat{x} = x_0 + \sum_{i=1}^{N} x_i \varepsilon_i \;\; \varepsilon_i \in [-1, 1]$
- Replace f by a linear approximation $\hat{f}(x) \approx \alpha x + \beta$
- $\gamma = \max_{x \in range(\hat{x})} |f(x) - \hat{f}(x)|$
- <2>$\hat{y} = f(\hat{x}) = \alpha x_0 + \beta + \sum_{i=1}^{N} \alpha x_i \varepsilon_i + \gamma \varepsilon_{N+1}$
- Each layer with width W adds W new coefficients.

Solution

Periodically replace a set of coefficients with a single new coefficients

$$condense(\hat{x}, \mathcal{D}) = x_0 + \sum_{i \notin \mathcal{D}} x_i \varepsilon_i + (\sum_{i \in \mathcal{D}} |x_i|) \varepsilon_{N+1}$$

Range bounds

Procedure 1 RangeBound($f_\theta, c, \{v_i\}$)

Input: A function $f_\theta : \mathbb{R}^d \to \mathbb{R}$ and a query box B of dimension $s \leq d$ defined by its center $c \in \mathbb{R}^d$, and s orthogonal box axis vectors $\{v_i \in \mathbb{R}^d\}$, not necessarily coordinate axis-aligned.

Output: A bound on the sign of $f_\theta(x)\ \ \forall x \in B$ as one of `POSITIVE`, `NEGATIVE`, or `UNKNOWN`.

$\hat{\mathbf{x}} \leftarrow c + \sum_{i=1}^{s} v_i \varepsilon_i$ ▷ *Construct affine bounds defining the box*

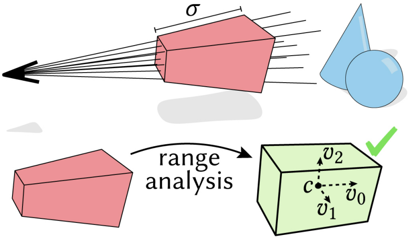
$\hat{\mathbf{y}} \leftarrow f_\theta(\hat{\mathbf{x}})$ ▷ *Propagate affine bounds (Section 3.2)*

$[y_-, y_+] \leftarrow \text{range}(\hat{\mathbf{y}})$ ▷ *Bound the output (Equation 3)*

if $y_- > 0$ **then return** `POSITIVE`

if $y_+ < 0$ **then return** `NEGATIVE`

else return `UNKNOWN`



[Sharp 2022]

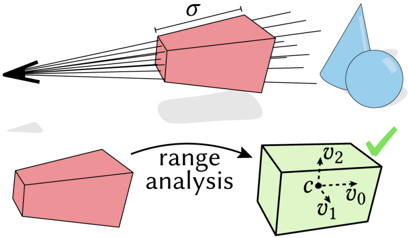
Range bounds

Procedure 1 RANGEBOUND($f_\theta, c, \{v_i\}$)

Input: A function $f_\theta : \mathbb{R}^d \to \mathbb{R}$ and a query box B of dimension $s \leq d$ defined by its center $c \in \mathbb{R}^d$, and s orthogonal box axis vectors $\{v_i \in \mathbb{R}^d\}$, not necessarily coordinate axis-aligned.

Output: A bound on the sign of $f_\theta(x) \ \forall x \in B$ as one of POSITIVE, NEGATIVE, or UNKNOWN.

1: $\hat{\mathbf{x}} \leftarrow c + \sum_{i=1}^{s} v_i \varepsilon_i$ ▷ *Construct affine bounds defining the box*
2: $\hat{\mathbf{y}} \leftarrow f_\theta(\hat{\mathbf{x}})$ ▷ *Propagate affine bounds (Section 3.2)*
3: $[y_-, y_+] \leftarrow \text{range}(\hat{\mathbf{y}})$ ▷ *Bound the output (Equation 3)*
4: **if** $y_- > 0$ **then return** POSITIVE
5: **if** $y_+ < 0$ **then return** NEGATIVE
6: **else return** UNKNOWN

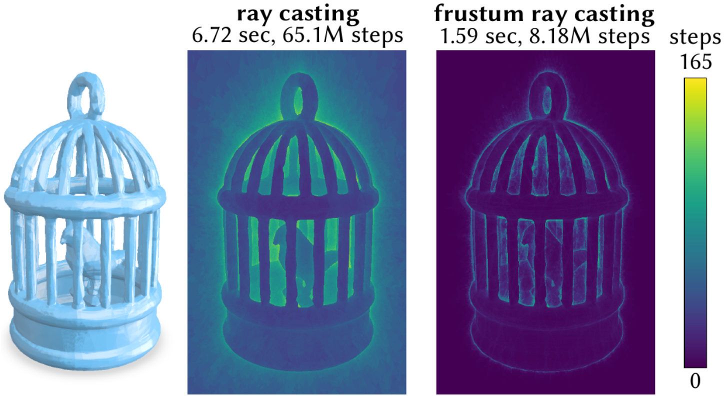


[Sharp 2022]

Unknown?

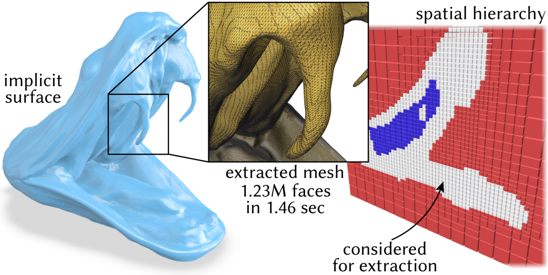
Subdivide the box.

Ray casting vs frustum ray casting

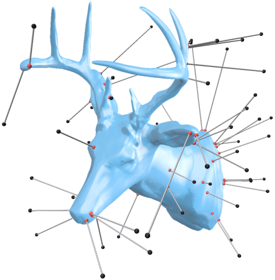


[Sharp 2022]

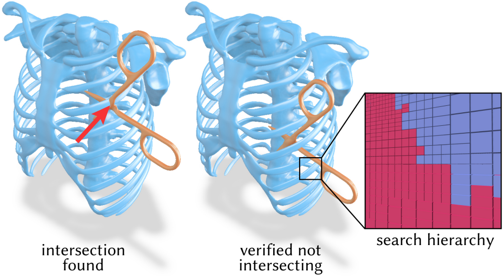
Applications



Mesh extraction



Closest point



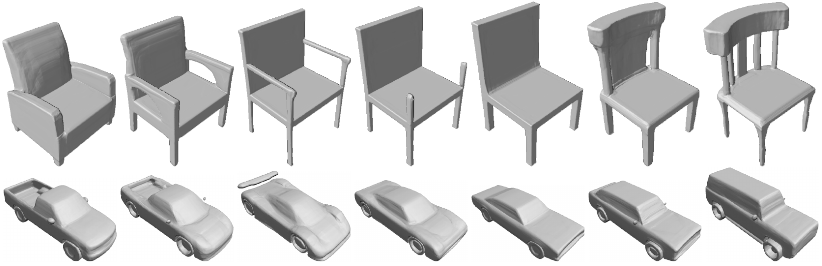
Mesh Intersection

Outline

Example-based shape reconstruction

- Deep SDF [Park 2019] learns a shape signature and deduces an implicit field (auto-decoder)
- Occupancy Network [Mescheder 2019] encoder-decoder to learn the occupancy (binary field).

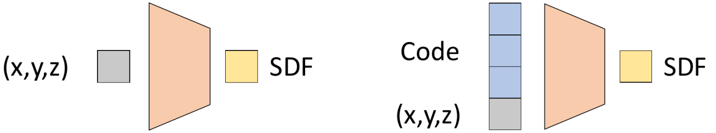
DeepSDF



[Park 2019]

- Represent an entire class of shapes in an implicit way

Training



(a) Single Shape DeepSDF **(b)** Coded Shape DeepSDF

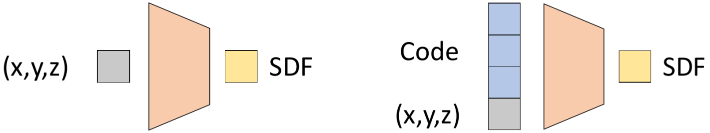
[Park 2019]

Single shape version

$$\mathcal{L}(f_\theta(x), s) = |clamp(f_\theta, \delta) - clamp(x, \delta)|$$

with $clamp(x, \delta) = \min(\delta, \max(-\delta, x))$, s isovalue.

Training



(a) Single Shape DeepSDF **(b)** Coded Shape DeepSDF

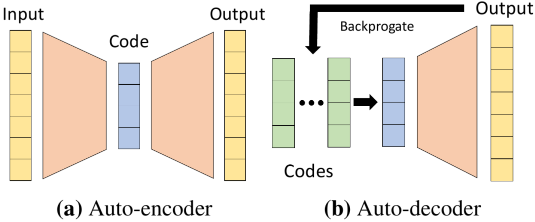
[Park 2019]

Latent shape version

$$f_\theta(z_i, x) = SDF^i(x)$$

Model several distance fields with a single network (factor in shape space)

Auto-decoder



[Park 2019]

- Usually: train an auto-encoder + throw away the encoder.
- Here: avoid spending computational resources on encoder.
- Handle shapes of different number of samples.

Model for the auto-decoder

- Data: N shapes $X_i = \{(x_j, s_j), s_j = SDF^i(x_j)\}$.
- Latent code z_i, prior $p(z_i)$ = centered Gaussian with spherical covariance.

$$p_\theta(z_i|X_i) = p(z_i) \prod_j p_\theta(s_j|z_i, x_j)$$

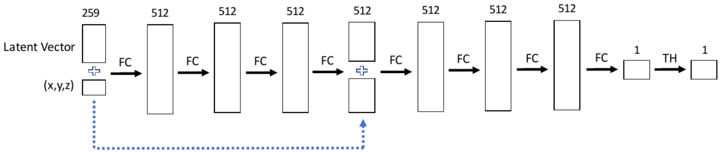
- Reformulation:

$$p(s_j|z_i, x_j) = \exp(-\mathcal{L}(f_\theta(z_i, x_j), s_j)) \text{ with } f_\theta \text{ an MLP.}$$

Training

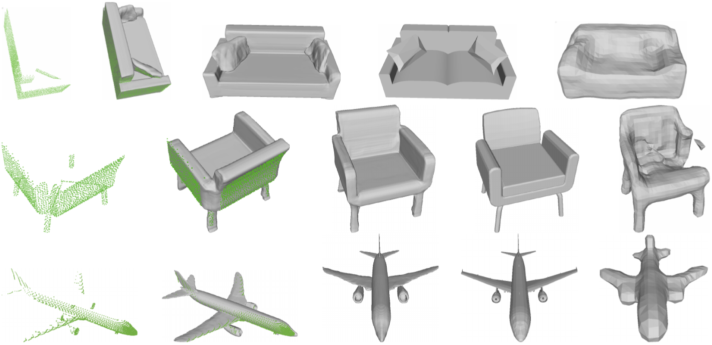
$$\text{argmin}_{\theta, \{z_i\}_{i=\mathbf{1}}^N} \sum_{i=1}^{N} \sum_{j=1}^{K} \mathcal{L}(f_\theta(z_i, x_j), s_j) + \frac{1}{\sigma^2} \|z_i\|_2^2$$

Network architecture



[park 2019]

results

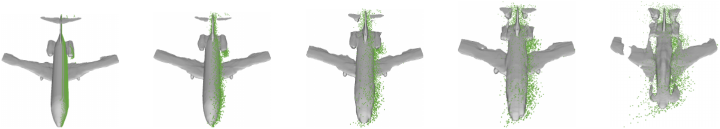


(a) Input Depth **(b)** Completion (ours) **(c)** Second View (ours) **(d)** Ground truth **(e)** 3D-EPN

[park 2019]

- solve for the shape code from partial shapes and reconstruct

results



(a) No noise (b) $\alpha = 0.01$ (c) $\alpha = 0.02$ (d) $\alpha = 0.03$ (e) $\alpha = 0.05$

[park 2019]

Conclusion

- Overview of Machine Learning methods
- Field changes every day!
- Some new tools useful even without a database