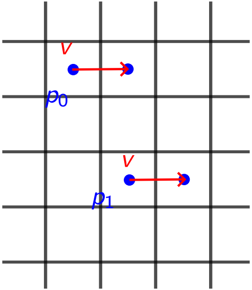
Implicit neural representations

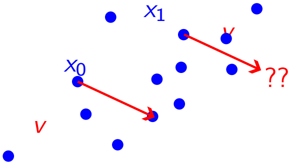
Julie Digne

Master MVA

November 6th 2024

Geometric data

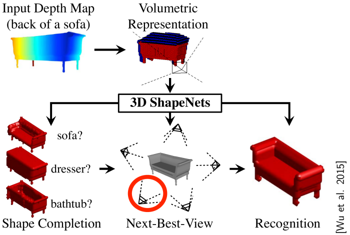


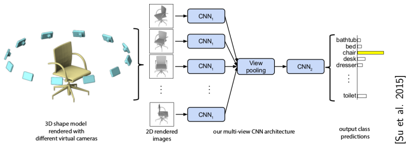


- No grid structure.
- Irregular Sampling, occlusions when scanning

Geometric Deep Learning

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



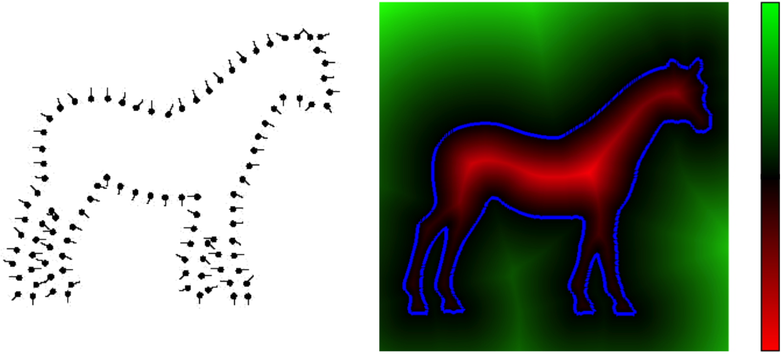


Today

Surfaces will be represented implicitly and we'll work on function estimation.

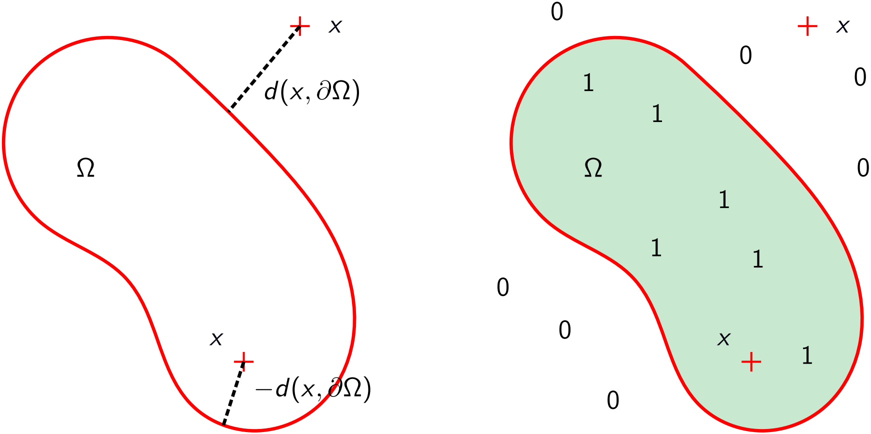
Outline

Implicit surface reconstruction - Principle



- See the surface as an isolevel of a given function
- Extract the surface by some contouring algorithm: Marching cubes [Lorensen Cline 87], Particle Systems [Levet et al. 06]

Implicit functions are not necessarily distance fields



Surface reconstruction from unorganized points [Hoppe et al. 92]

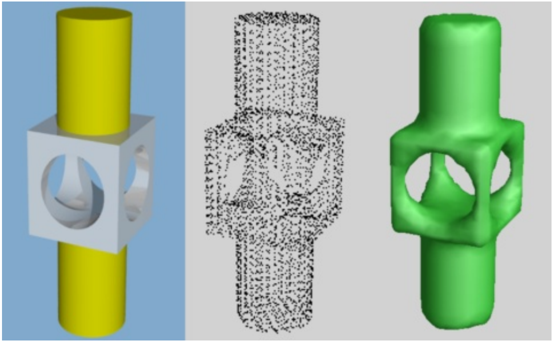
- Input: a set of 3D points
- *Idea:* for points on the surface the signed distance transform has a gradient equal to the normal

$$F(p) = \pm \min_{q \in \mathcal{S}} \|p - q\|$$

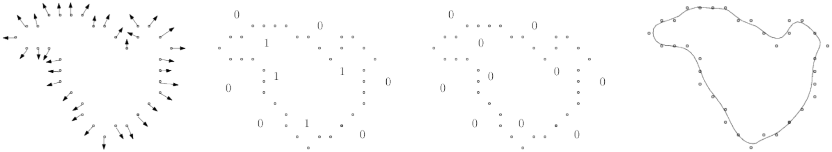
- 0 is a regular value for F and thus the isolevel extraction will give a manifold
- Compute an associated tangent plane (o_i, n_i) for each point p_i of the point set
- Orientation of the tangent planes as explained before.

Surface reconstruction from unorganized points [Hoppe et al. 92]

- Once the points are oriented
- For each point p, find the closest centroid o_i
- Estimated signed distance function: $\hat{f}(p) = n_i \cdot (p - o_i)$



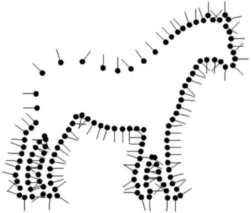
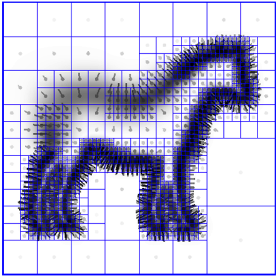
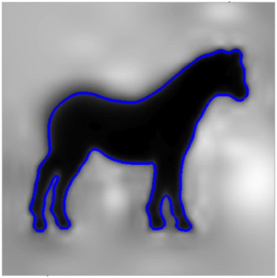
Poisson Surface Reconstruction [Kazhdan et al. 2006]



- Input: a set of oriented samples
- Reconstructs the indicator function of the surface and then extracts the boundary.
- Trick: Normals sample the function's gradients

Poisson Surface Reconstruction [Kazhdan et al. 2006]

1. Transform samples into a vector field
2. Fit a scalar-field to the gradients
3. Extract the isosurface

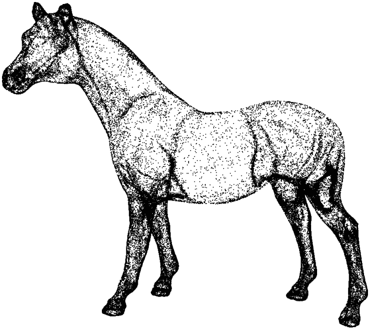
Poisson Surface Reconstruction [Kazhdan et al. 2006]

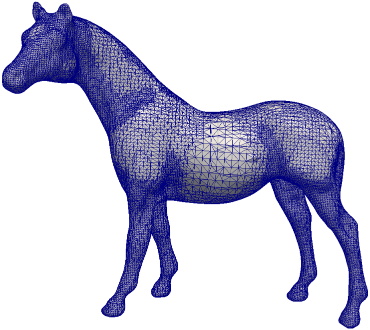
- To fit a scalar field χ to gradients $\vec{V}$, solve:

$$\min_{\chi} \|\nabla\chi - \vec{V}\|$$

- Eq to:

$$\nabla \cdot (\nabla\chi) - \nabla \cdot \vec{V} = 0 \Leftrightarrow \Delta\chi = \nabla \cdot \vec{V}$$





- Gradient Function of an indicator function = unbounded values on the surface boundaries
- We use a smoothed indicator function

Lemma

The gradient of the smoothed indicator function is equal to the smoothed normal surface field.

$$\nabla \cdot (\chi \star \tilde{F})(q_0) = \int_{\partial M} \tilde{F}(q_0 - p) \cdot \vec{N}_{\partial M}(p) dp$$

Chicken and Egg problem: to compute the gradient one must be able to compute an integral over the surface!!

- Approximate the integral by a discrete summation
- Surface partition in patches $\mathcal{P}(s)$:

$$\nabla \cdot (\chi \star \tilde{F})(q_0) = \sum_s \int_{\mathcal{P}(s)} \tilde{F}(q_0 - p) \cdot \vec{N}_{\partial M}(p) dp$$

- Approximation on each patch:

$$\nabla \cdot (\chi \star \tilde{F})(q_0) = \sum_s |\mathcal{P}(s)| \tilde{F}(q_0 - s) \cdot \vec{N}(s)$$

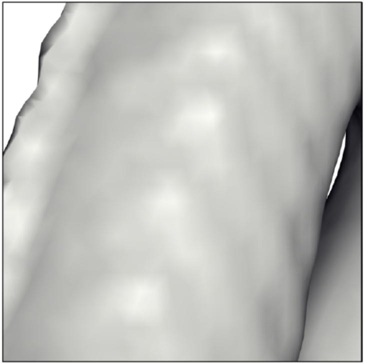
- Let us define $V(q_0) = \sum_s |\mathcal{P}(s)| \tilde{F}(q_0 - s) \cdot \vec{N}(s)$

Problem Discretization

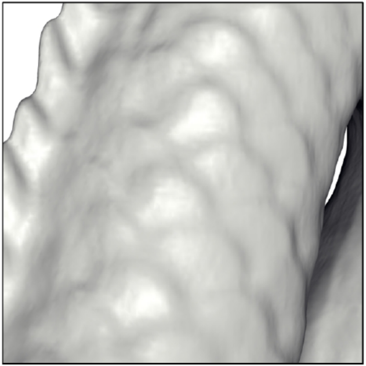
- Build an adaptive octree $\mathcal{O}$
- Associate a function F_o to each node o of $\mathcal{O}$ so that: $F_o(q) = F(\frac{q-o.c}{o.w})\frac{1}{o.w^3}$ ($o.c$ and $o.w$ are the center and width of node o).$\Rightarrow$ multiresolution structure
- The base function F is the *nth* convolution of a box filter with itself
- $$\vec{V}(q) = \sum_{s \in S} \sum_{o \in \mathcal{N}(s)} \alpha_{o,s} F_o(q) s.\vec{N}$$
- Look for χ such that its projection on $span(F_o)$ is closest to ∇V :
- Minimize $\sum_{o \in \mathcal{O}} \langle \Delta\chi - \nabla \cdot V, F_o \rangle^2$
- Extracted isovalue: mean value of χ at the sample positions

Varying octree depth

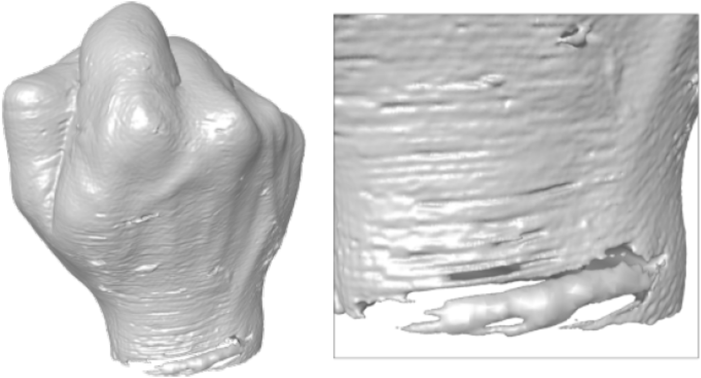
Varying octree depth



Varying octree depth



Resilience to bad normals



Mullen et al. 2010

Moving Least Squares surfaces

definition

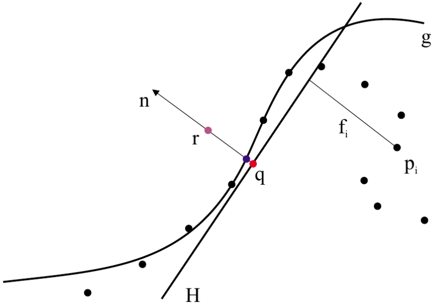
A set of points $(x_i) \in \mathbb{R}^3$ with associated function values f_i, Moving least squares approximation

$$p(x) = argmin_y \sum_i (y - f_i)^2 \theta(\|x - x_i\|)$$

with θ a decreasing function (e.g. $\theta(t) = \exp -t^2)$)

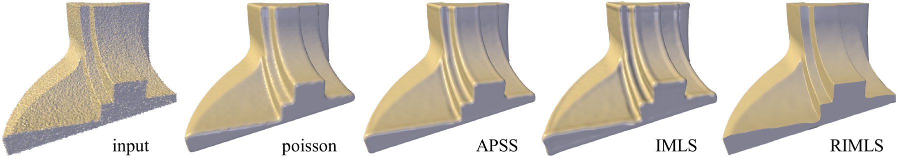
Adaptation to 3D surfaces

- For each point compute its projection on the surface. The Point Set Surface is defined as the fixed points of this projection procedure.
- Variants: APSS [Guennebaud 2007], RIMLS [Oztireli 2009]
- Can be used to define a distance to a surface (+surface reconstruction via marching cubes).



[Alexa 2001]

Results



[Oztireli 2009]

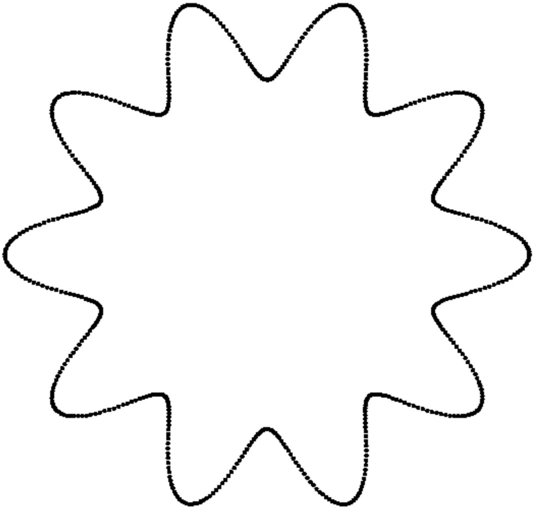
From the signed distance function to the mesh

- At each point in $\mathbb{R}^3$, the signed distance function to the surface can be estimated
- Extract the 0 levelset of this function: points where this function is 0

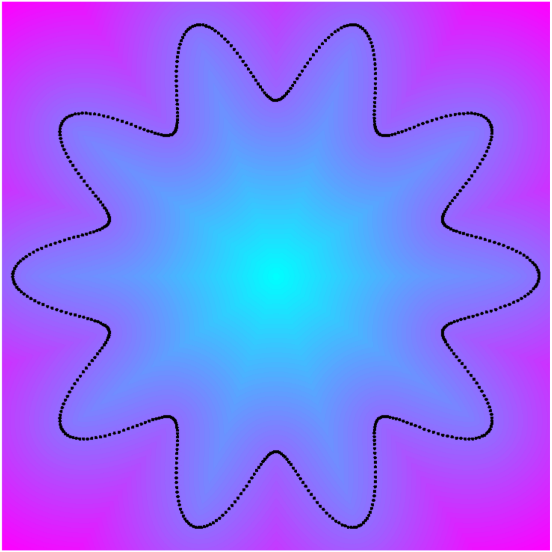
Approximation

Evaluate the function at the vertices of a grid and deduce the local geometry of the surface in each grid cube.

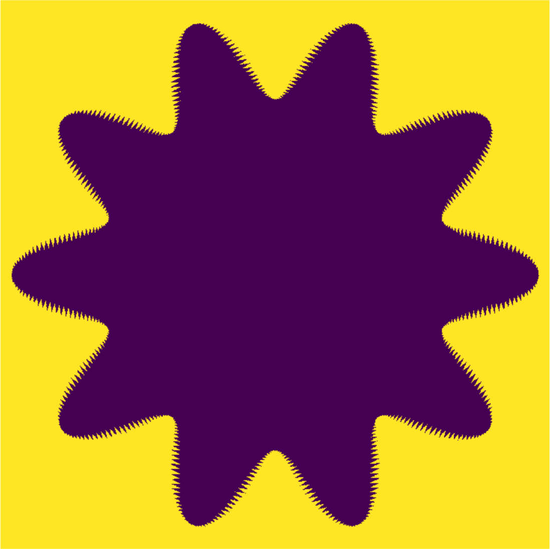
Example in 2D



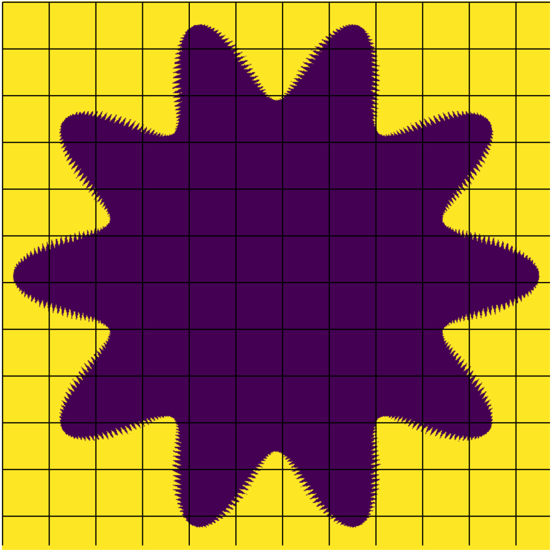
Example in 2D



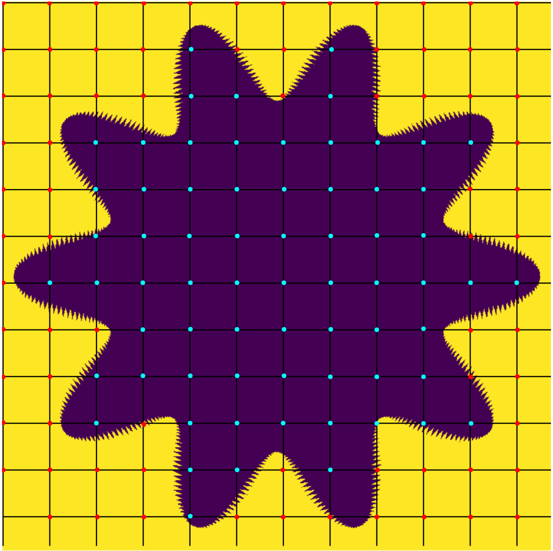
Example in 2D



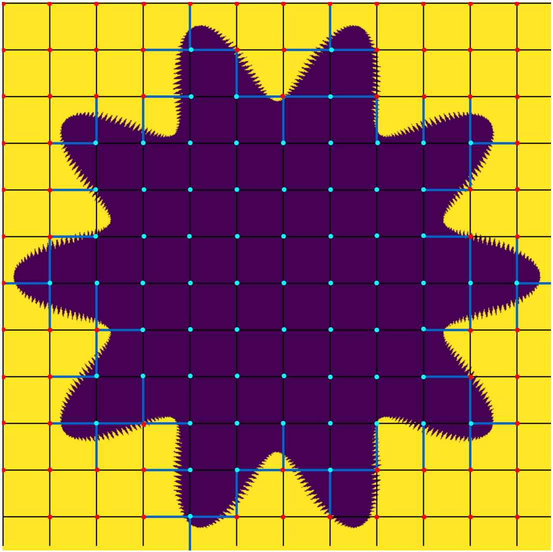
Example in 2D



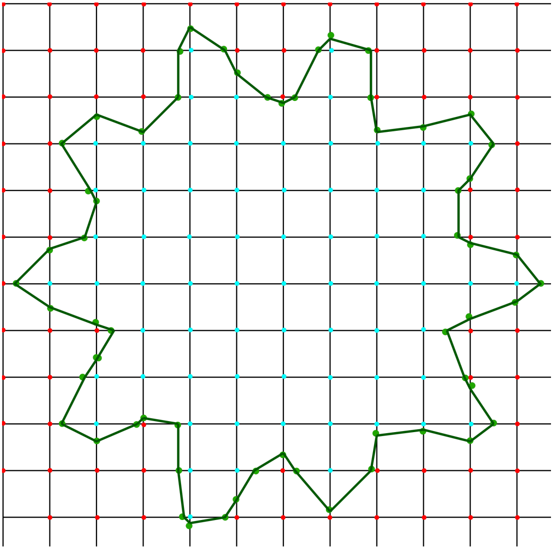
Example in 2D



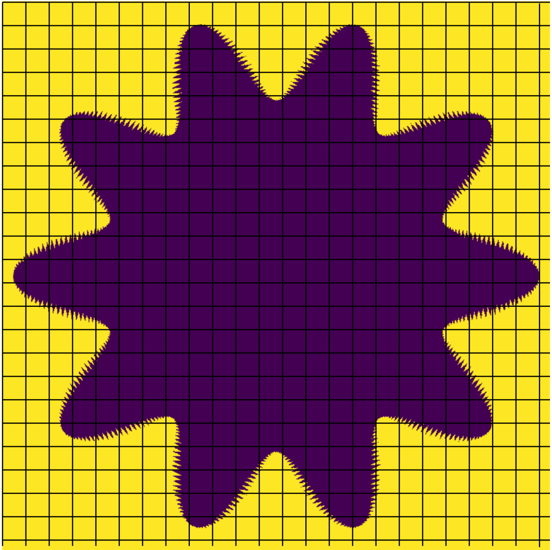
Example in 2D



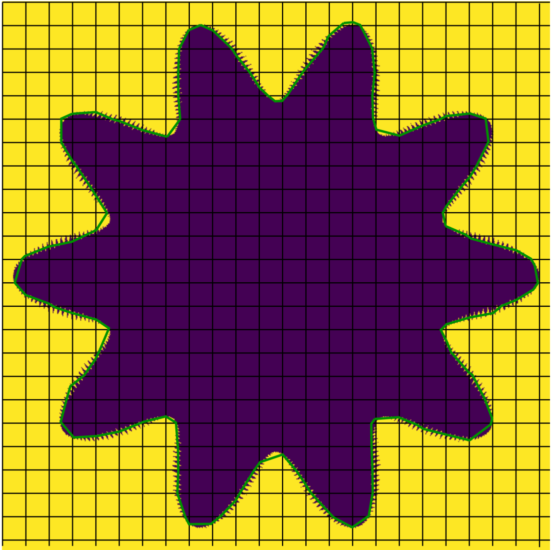
Example in 2D



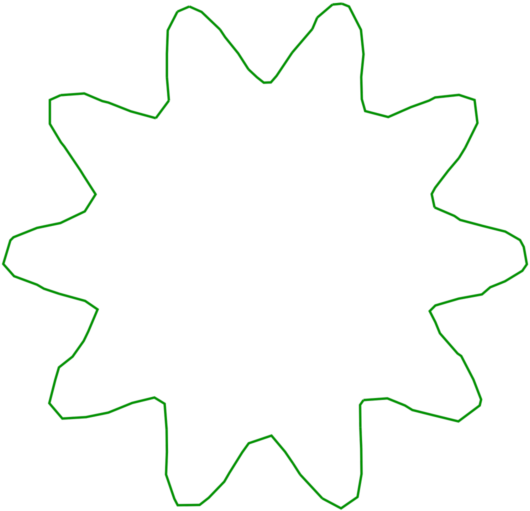
Example in 2D



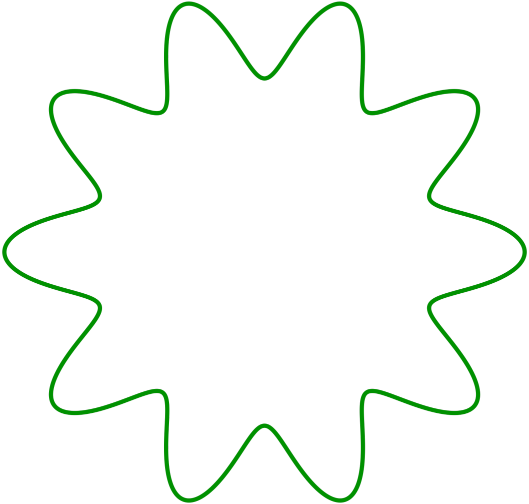
Example in 2D



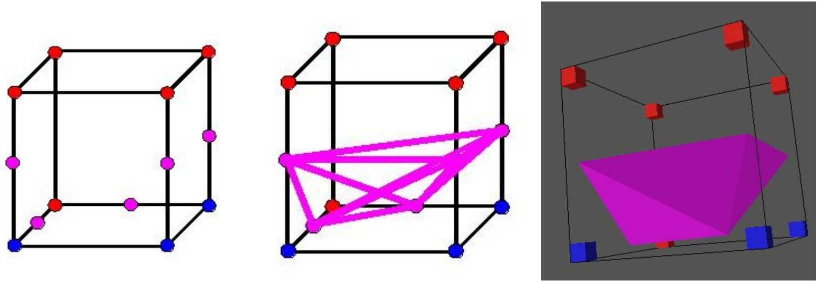
Example in 2D



Example in 2D



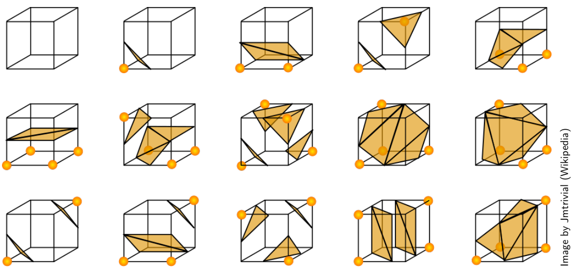
From Marching Squares to Marching Cubes



Images by Ben Anderson

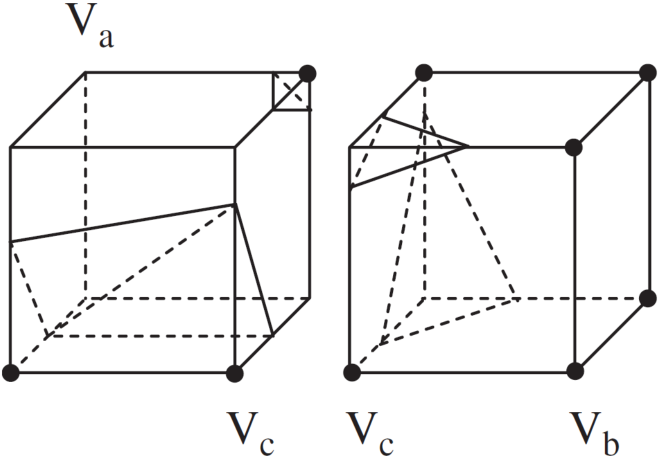
Drawing lines between intersection points is ambiguous and does not give a surface patch.

Look-up tables

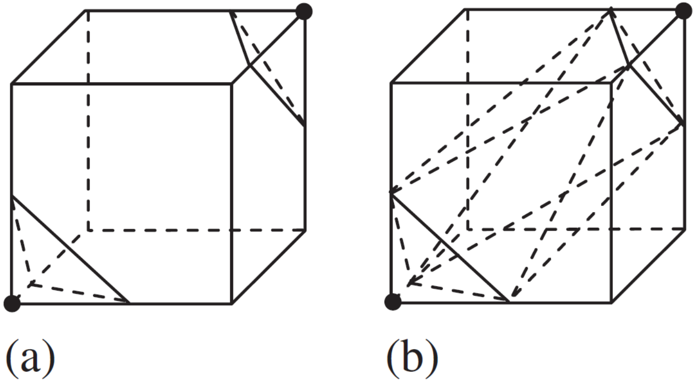


- There are $2^8 = 256$ possible cases for cube corner values.
- By symmetry + rotation arguments it reduces to 15 cases.
- Build a look-up table giving the grid cell triangulation based on the corner values case.

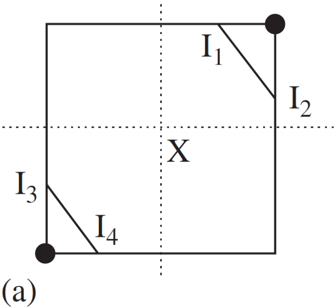
Ambiguous cases



Ambiguous cases



Ambiguous cases



- Refine the grid to remove ambiguation
- Switch to marching tetrahedra algorithm

Advantages and drawbacks of the Implicit surface reconstruction methods

Drawbacks

- Only semi-sharp, loss of details
- Final mesh not interpolating the initial pointset
- Marching cubes introduces artefacts
- Watertight surface, very bad with open boundaries

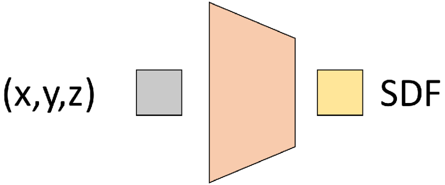
Advantages

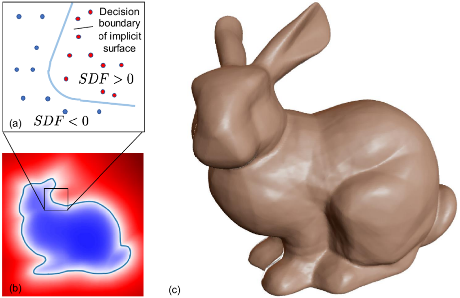
- Noise robustness
- Watertight surface, hole closure
- Most standard way of reconstructing a surface

Outline

Learning a signed distance [DeepSDF - Park et al. 2019]

- Learn a SDF u_θ to a shape $\mathcal{X}$, knowing a set of points $x_i \in X$.





Learning

- Input data: a set of points y_i in $\mathbb{R}^3$ and their distance to the surface $s_i = SDF(y_i)$

Loss function

$$\mathcal{L}(\theta) = \sum_i |clamp(u_\theta(y_i), \delta) - clamp(s_i, \delta)|$$

with $clamp(h, \delta) = \min(\delta, \max(-\delta, h))$.

- δ controls the width of the region of interest around the surface. In practice $\delta = 0.1$.

Learning

- Input data: a set of points y_i in $\mathbb{R}^3$ and their distance to the surface $s_i = SDF(y_i)$

Loss function

$$\mathcal{L}(\theta) = \sum_i |clamp(u_\theta(y_i), \delta) - clamp(s_i, \delta)|$$

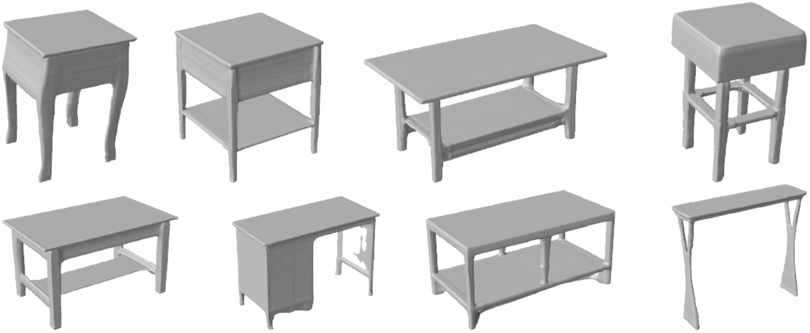
with $clamp(h, \delta) = \min(\delta, \max(-\delta, h))$.

- δ controls the width of the region of interest around the surface. In practice $\delta = 0.1$.

Architecture

8 layers MLP, (width 512), dropout, ReLU activation function (tanh for the last layer) + weight normalization.

Results



[Park 2019]

Learning an occupancy function [Mescheder 2019]

Occupancy function

Given an object as a compact subset $\Omega \subset \mathbb{R}^3$, the occupancy function $u : \mathbb{R}^3 \to 0, 1$ is such that:

$$u_\theta(x) = \begin{cases} 1 & \text{if } x \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

- Neural network will learn a function $u_\theta(x)$ predicting whether u is inside Ω or outside Ω

Losses

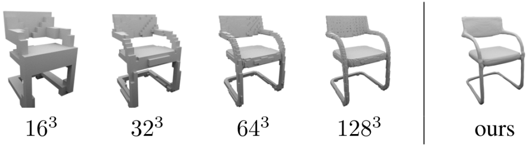
- Input data: a set of points y_i in $\mathbb{R}^3$ and their positions relatively to the surface $o_i = 0$ or 1.

Loss function

$$\mathcal{L}(\theta) = \sum_i BCE(u_\theta(y_i), o_i)$$

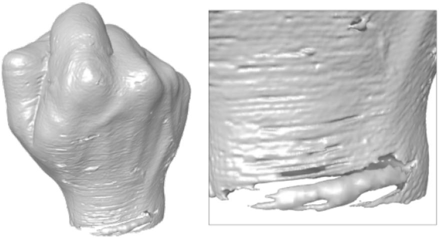
- This is the single shape loss. Occupancy networks are mostly used in the context of latent shape spaces, see next course for more details!

Results



[Mescheder 2019]

Learning an unsigned distance

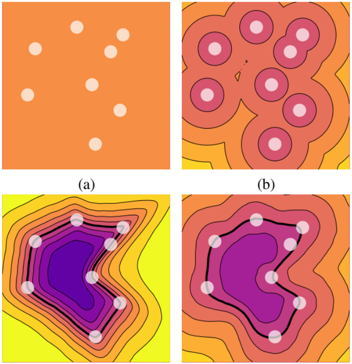


Mullen et al. 2010

- Normal direction is *easy* to compute
- *Consistent* normal orientation is hard to compute
- Bad normal orientations create artifacts for the SDF estimation

Sign agnostic distance function (Aatzmon 2020]

- Unoriented points (not even using normal direction)
- Signed distance function or surface indicator function


(a) (b)
(c) (d)

Losses

Loss function

$$loss(\theta) = \mathbb{E}_{x \in \mathcal{D}_X}[\tau(|u_\theta(x)|, h_X(x))]$$

- $\mathcal{D}_X$ is a distribution of points
- τ is a similarity function.
- h_X is an unsigned distance to the shape.

Losses

Loss function

$$loss(\theta) = \mathbb{E}_{x \in \mathcal{D}_X}[\tau(|u_\theta(x)|, h_X(x))]$$

- $\mathcal{D}_X$ is a distribution of points
- τ is a similarity function.
- h_X is an unsigned distance to the shape.

Conditions on τ

$\tau : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$ is such that:

- Sign agnostic: $\tau(-a, b) = \tau(a, b) \forall (a, b) \in \mathbb{R} \times \mathbb{R}^+$
- Monotonicity: $\frac{\partial \tau}{\partial a} = \rho(a - b) \forall (a, b) \in \mathbb{R}^+ \times \mathbb{R}$

Useful for the theorems guaranteeing reconstruction properties.

Choice of h_X and similarity τ

- ℓ^2 distance:

$$h_2(y) = \min_{x \in X} \|y - x\|_2$$

- ℓ^0 distance:

$$h_0(y) = \begin{cases} 1 \text{ if } y \in X \\ 0 \text{ otherwise.} \end{cases}$$

- $\tau(a, b) = ||a| - b|^l$

Choice of h_X and similarity τ

- ℓ^2 distance: Signed distance function

$$h_2(y) = \min_{x \in X} \|y - x\|_2$$

- ℓ^0 distance: indicator of the surface

$$h_0(y) = \begin{cases} 1 \text{ if } y \in X \\ 0 \text{ otherwise.} \end{cases}$$

- $\tau(a, b) = ||a| - b|^l$

Choice of point distribution $\mathcal{D}_X$

- Data points $X = x$, not enough to learn the whole field
- For the ℓ^2 distance:

$$\mathcal{D}_X = \sum_i \mathcal{N}(x_i, \sigma^2 I)$$

$$\mathcal{L}_2(\theta) = \mathbb{E}_{y \sim \mathcal{D}_X}[|u_\theta(y)| - h_2(y)]$$

- For the ℓ^0 distance:

$$\mathcal{D}_X = \sum_i \mathcal{N}(x_i, \sigma^2 I) + \sum_i \delta_{x_i}$$

$$\mathcal{L}_0(\theta) = \mathbb{E}_{y \sim \sum_i \mathcal{N}(x_i, \sigma^2 I)}[|u_\theta(y)| - 1] + \mathbb{E}_{y \sim \sum_i \delta_{x_i}}[|u_\theta(y)|]$$

Neural Architecture

MLP Architecture

$$u_\theta(x) = \varphi(w^T f_l \circ f_{l-1} \circ \cdots \circ f_1(x) + b) + c$$

with:

$$f_i(x) = \nu(W_i x + b_i)$$

$b_i \in \mathbb{R}^{d_i^{out}}$, $W_i \in \mathbb{R}^{d_i^{out} \times d_i^{in}}$, $w \in \mathbb{R}^{d_l^{out}}$ and $c \in \mathbb{R}$. ν are ReLU activation functions and φ a strong nonlinearity activation function.
+ Skip connection to the middle layer.

Neural Architecture

MLP Architecture

$$u_\theta(x) = \varphi(w^T f_l \circ f_{l-1} \circ \cdots \circ f_1(x) + b) + c$$

with:

$$f_i(x) = \nu(W_i x + b_i)$$

$b_i \in \mathbb{R}^{d_i^{out}}$, $W_i \in \mathbb{R}^{d_i^{out} \times d_i^{in}}$, $w \in \mathbb{R}^{d_l^{out}}$ and $c \in \mathbb{R}$. ν are ReLU activation functions and φ a strong nonlinearity activation function.
+ Skip connection to the middle layer.

Strong activation

$\varphi : \mathbb{R} \leftarrow \mathbb{R}$ is called a strong non-linearity if it is differentiable almost everywhere, antisymmetric: $\varphi(a) = -\varphi(-a)$ and there exists $\beta \in \mathbb{R}^+$ so that $\frac{1}{\beta} \geq \varphi'(a) \geq \beta$ for all $a \in \mathbb{R}$ where it is defined.

Neural Architecture

MLP Architecture

$$u_\theta(x) = \varphi(w^T f_l \circ f_{l-1} \circ \cdots \circ f_1(x) + b) + c$$

with:

$$f_i(x) = \nu(W_i x + b_i)$$

$b_i \in \mathbb{R}^{d_i^{out}}$, $W_i \in \mathbb{R}^{d_i^{out} \times d_i^{in}}$, $w \in \mathbb{R}^{d_l^{out}}$ and $c \in \mathbb{R}$. ν are ReLU activation functions and φ a strong nonlinearity activation function.
+ Skip connection to the middle layer.

Strong activation

$\varphi : \mathbb{R} \leftarrow \mathbb{R}$ is called a strong non-linearity if it is differentiable almost everywhere, antisymmetric: $\varphi(a) = -\varphi(-a)$ and there exists $\beta \in \mathbb{R}^+$ so that $\frac{1}{\beta} \geq \varphi'(a) \geq \beta$ for all $a \in \mathbb{R}$ where it is defined.

- In practice we take $\varphi(a) = a$ or $\varphi(a) = \tanh(a) + \gamma a$ with $\gamma \geq 0$.

Initialization

- Why? Avoid some local minima.

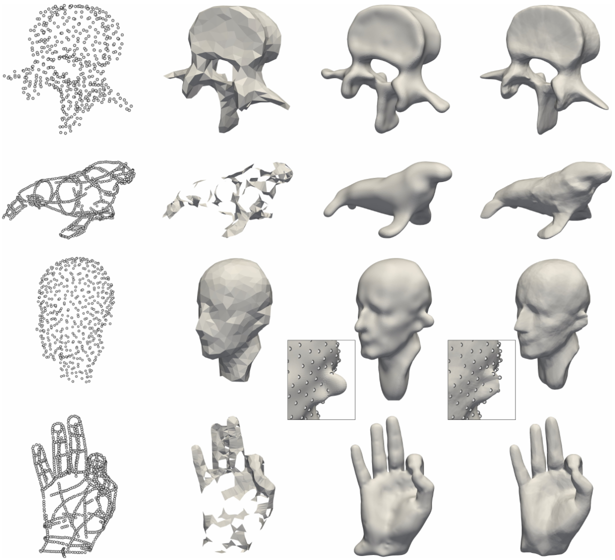
Theorem

Let u_θ be an MLP, Let $b_i = 0$, and W_i iid for a normal distribution $\mathcal{N}(0, \frac{\sqrt{2}}{\sqrt{d_i^{out}}})$ $1 \leq i \leq l$, $w = \frac{\sqrt{\pi}}{\sqrt{d_l^{out}}}1$, $c = -r$ then: $u_\theta(x) = \phi(\|x\| - r)$.

Properties

- *Plane reproduction:* If data points lie on a hyperplane, this plane is a critical point of the loss.
- *Local plane reconstruction*: Can be applied locally for surfaces.

Results



Input point cloud, Ball Pivoting, Variational implicit reconstruction, SAL

Outline

Implicit neural field

- 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

- 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:

$$\left\{ \begin{array}{rl} \|\nabla u\| & = 1 \\ u_{|\partial\Omega} & = 0 \\ \nabla u_{|\partial\Omega]} & = \mathrm{n} \end{array} \right. \tag{1}$$

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

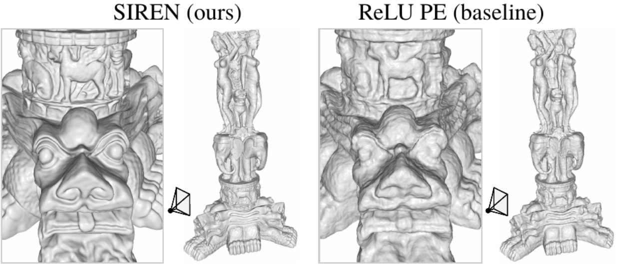


Figure 4: A comparison of SIREN used to fit a SDF from an oriented point clouse against the same fitting performed by an MLP using a ReLU PE (proposed in [35]).

From [Sitzmann 2020]

Periodic Activation Functions [Sitzmann 2021]

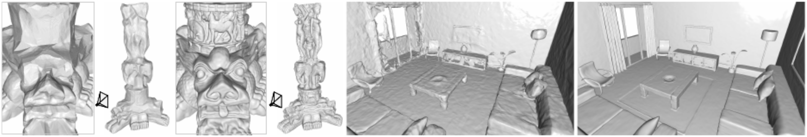
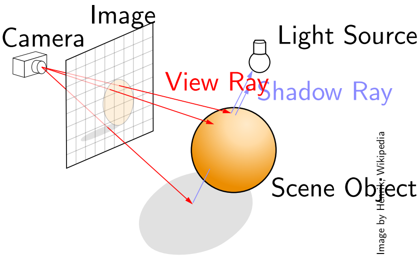


Figure 4: Shape representation. We fit signed distance functions parameterized by implicit neural representations directly on point clouds. Compared to ReLU implicit representations, our periodic activations significantly improve detail of objects (left) and complexity of entire scenes (right).

From [Sitzmann 2020]

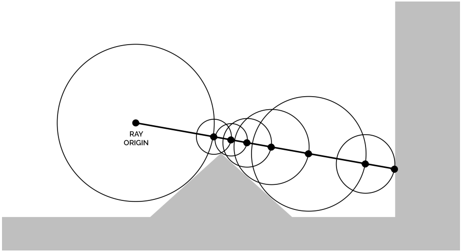
Outline

Sphere tracing



- Requires to compute ray/surface intersection.
- Direct intersection with explicit representations (Meshes/Geometric primitives)

Sphere tracing [Hart 1996]



1. Input: a point x and direction v, a signed distance field u.
2. Initialize $t = 0$
3. While $t < D$
   1. $x_t = x + t\text{v}$
   2. $d = u(x_t)$
   3. If $d < \varepsilon$ Return x_t
   4. Else Increment $t = t + d$

After intersection...

- Similar to ray tracing, rebounds can be computed
- Direct light only: color = scalar product of normal at intersection point and light direction.

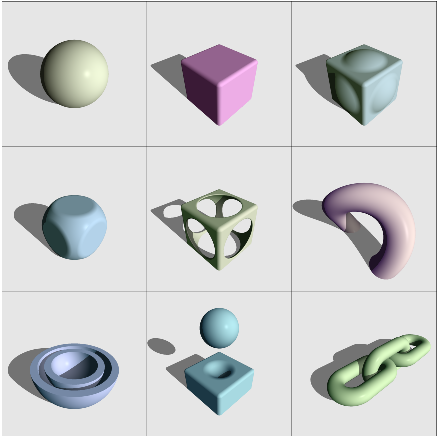
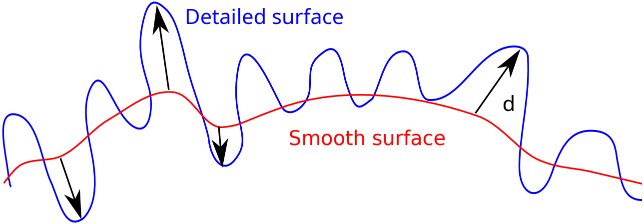


Image by Hiroki Sakuma

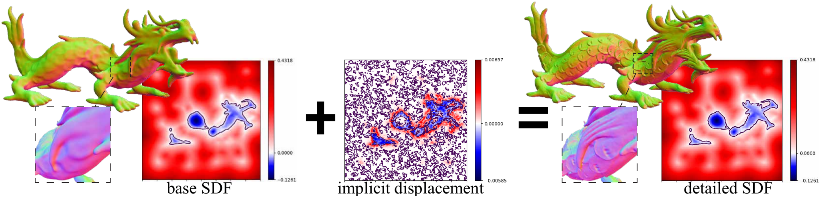
Outline

Implicit displacement field [Yifan 2022]



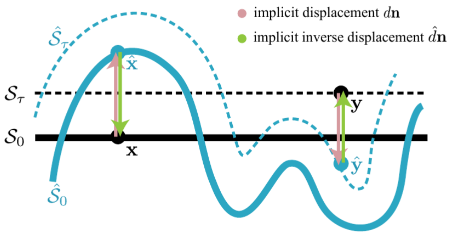
- Decompose the surface into a smooth base and a displacement field
- *Both* the smooth surface and the displacement field are learned

Overview



[Yifan 2022]

Implicit displacement field - definition



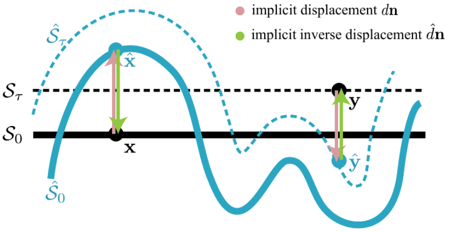
[Yifan 2022]

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

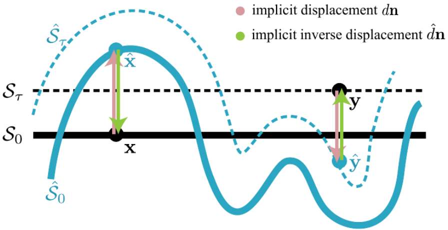
Smooth base SDF f, detailed SDF $\hat{f}$, an implicit displacement field (IDF)

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

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

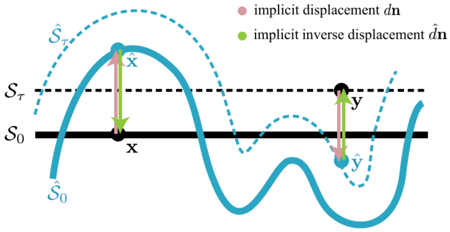


[Yifan 2022]

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

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

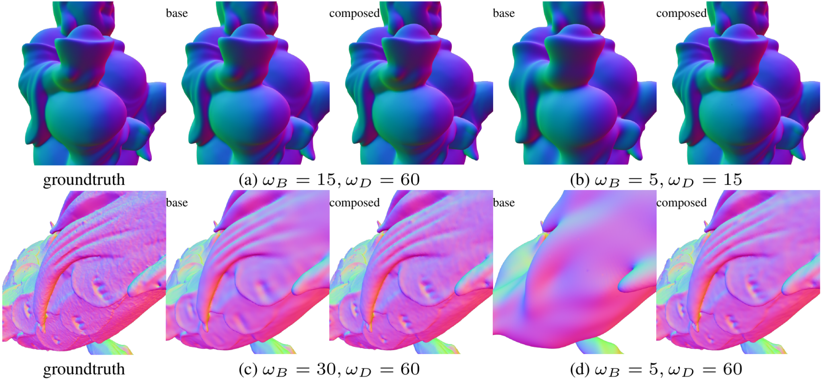
$$\hat{f}(x) = \mathcal{N}_{\omega_B}(x + \chi(f(x))\mathcal{N}_{\omega_D}(x)\frac{\nabla f(x)}{\|\nabla f(x)\|})$$

where χ is an attenuation function

Loss

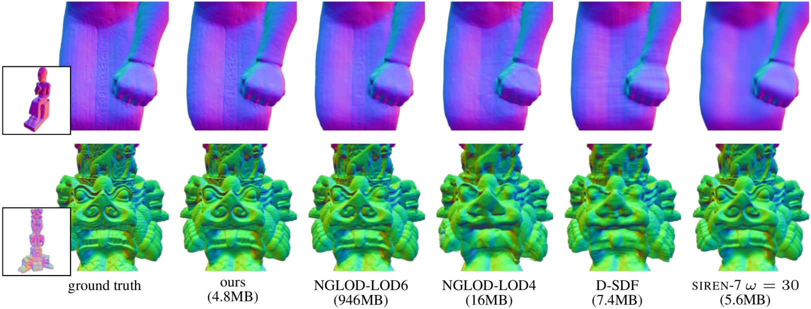
$$\mathcal{L}_{\hat{f}} = \sum_{x \in \mathbb{R}^3} \lambda_0 |\|\nabla \hat{f}(x)\| - 1| + \sum_{(p,n) \in \partial\Omega} (\lambda_1 |\hat{f}(p)| + \lambda_2 (1 - \langle \nabla \hat{f}(p), n \rangle)) + \sum_{x \in \mathbb{R}^3} \lambda_3 \exp(-100 \hat{f}(x))$$

Results - Surface decomposition



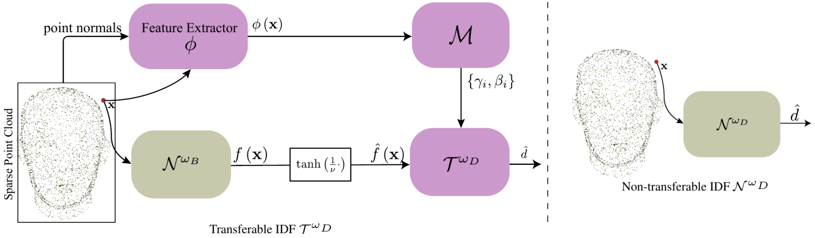
[Yifan 2022]

Detailed surface reconstruction



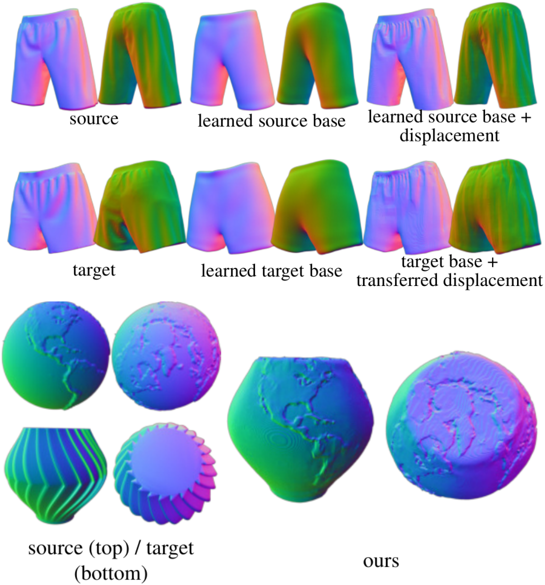
[Yifan 2022]

Detail transfer



[Yifan 2022]

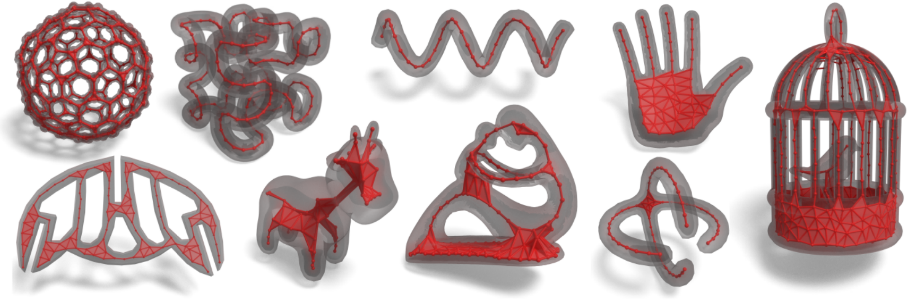
Detail transfer results



[Yifan 2022]

Outline

Regularizing INR away from the surface

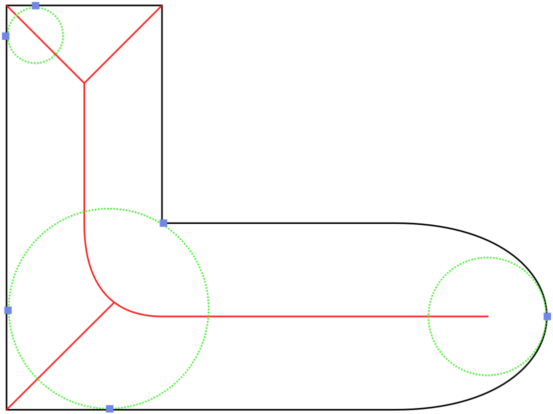


[Clémot, Digne 2023]

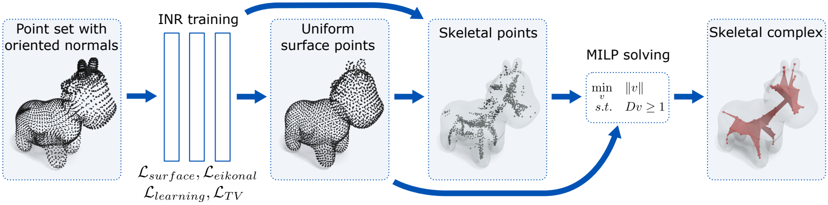
Medial Axis

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



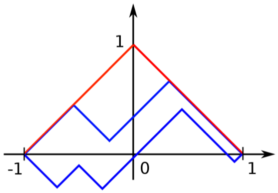
Point set with oriented normals
INR training
$\mathcal{L}_{surface}, \mathcal{L}_{eikonal}$
$\mathcal{L}_{learning}, \mathcal{L}_{TV}$
Uniform surface points
Skeletal points
MILP solving
$\min_{v} \ \|v\|$
$s.t. \ Dv \geq 1$
Skeletal complex

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



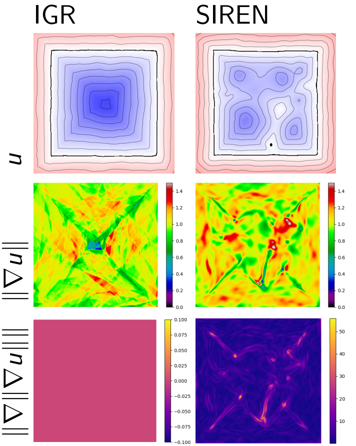
[Camilli 2014]

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:

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

- Surface loss:

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

- Learning point loss

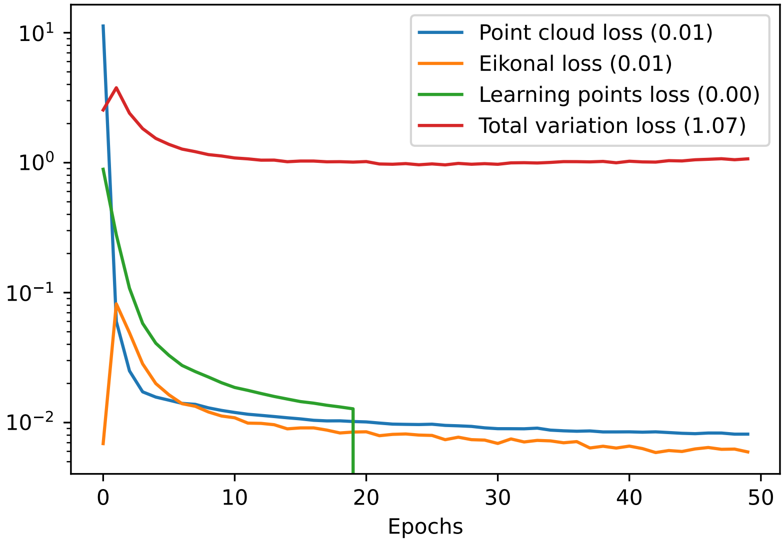
$$\mathcal{L}_{\text{learning}} = \sum_{p \in \mathcal{P}} (u(p) - d(p))^2 + \sum_{p \in \mathcal{P}} 1 - \frac{\nabla u(p) \cdot \nabla d(p)}{\|\nabla u(p)\| \, \|\nabla d(p)\|} \tag{4}$$

- \+ TV loss

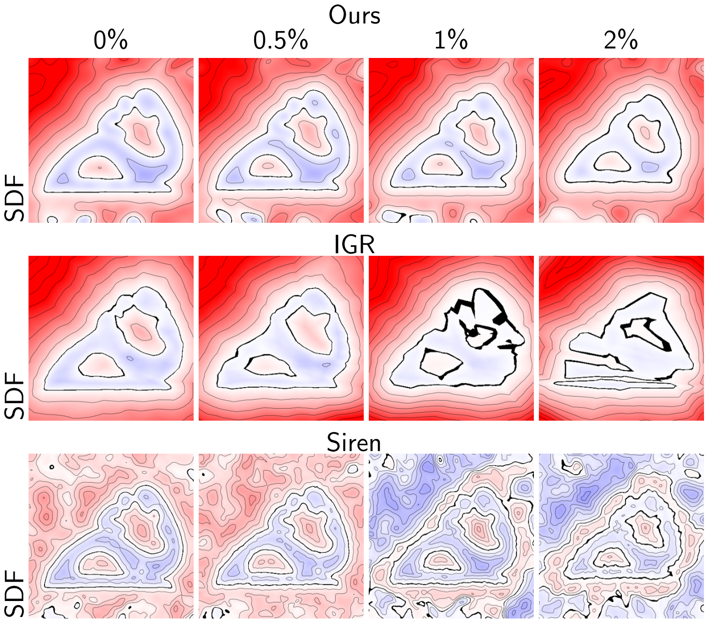
Loss

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

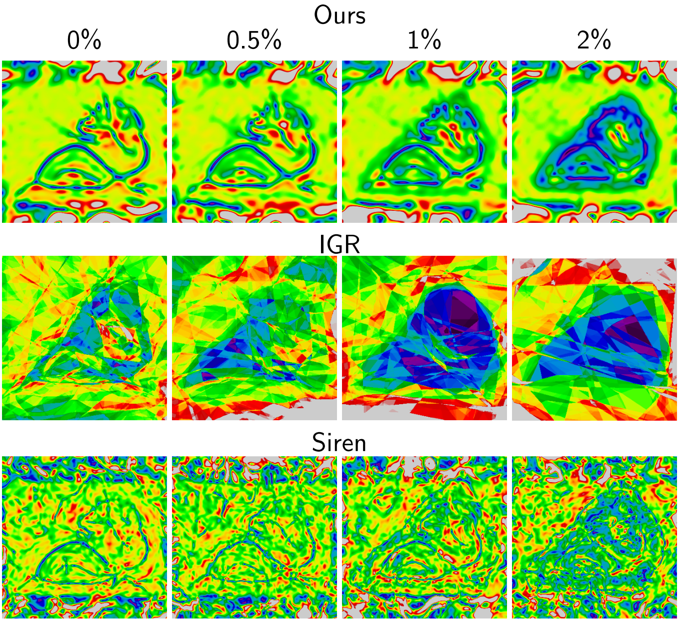
Convergence



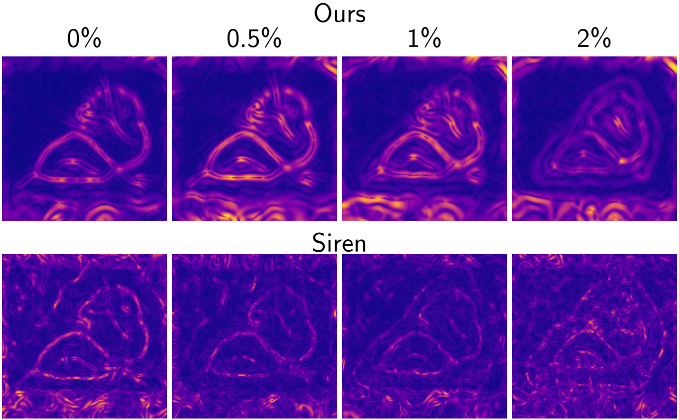
Resulting Fields



$\|\nabla u\|$



$\nabla \|\nabla u\|$



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 $(N \times M)$

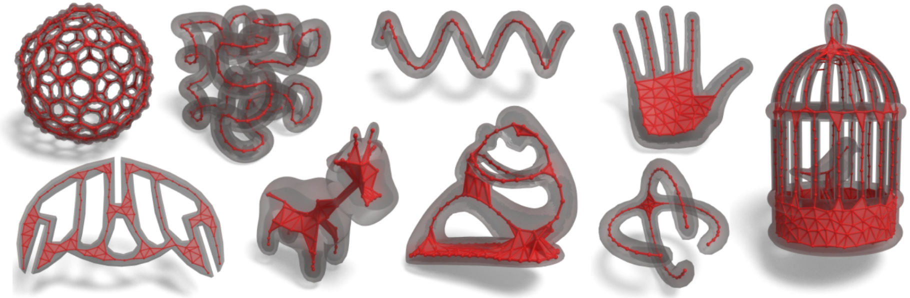
$$D_{ij} = 1 \text{ if } \|p_i - s_j\| - r_j \leq \delta \text{ and } 0 \text{ otherwise}$$

  - Mixed Integer Linear Problem:

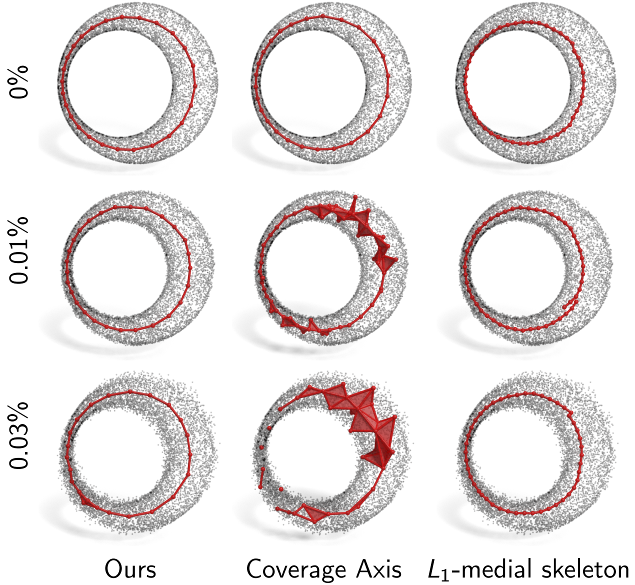
$$\begin{aligned} \min \quad & \|v\|_2 \\ \text{s.t.} \quad & Dv \succeq 1 \end{aligned} \tag{6}$$

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

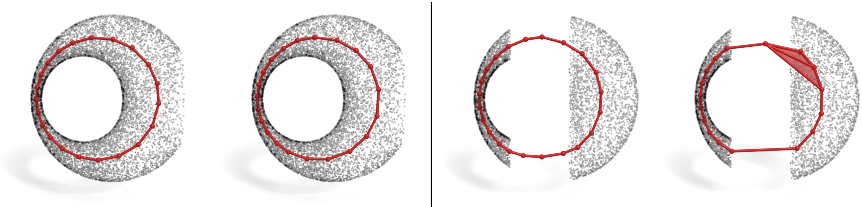
Results



Results

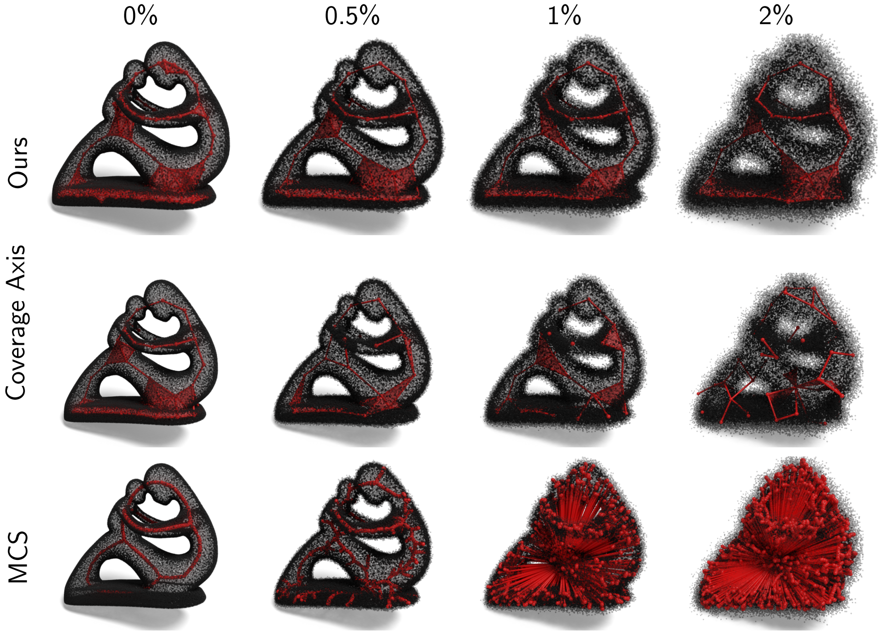


Results

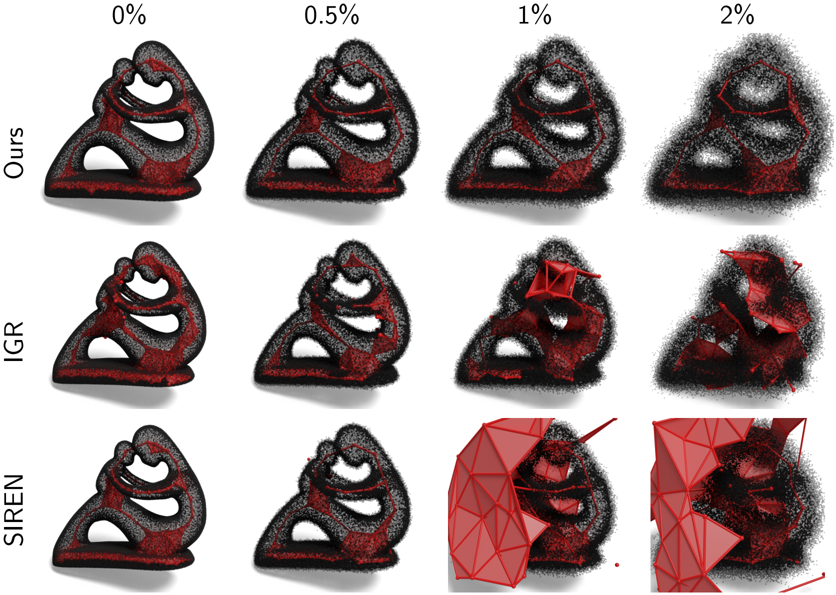


Ours | Coverage Axis | Ours | Coverage Axis

With noise



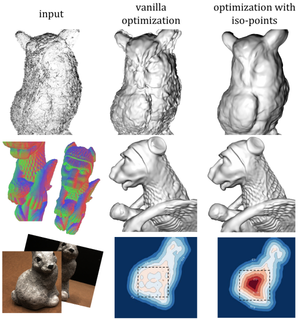
With noise



Outline

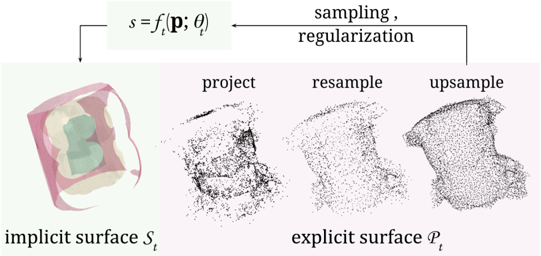
Projecting points on the surface [Yifan 2021]

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



[Yifan 2021]

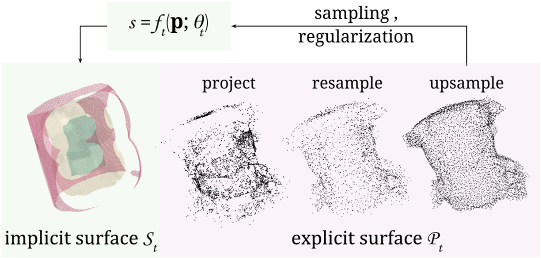
Projection on the surface



[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

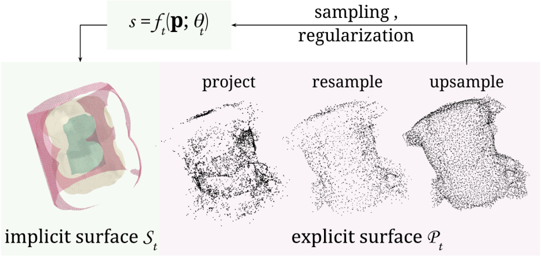
Uniform resampling



[Yifan 2021]

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

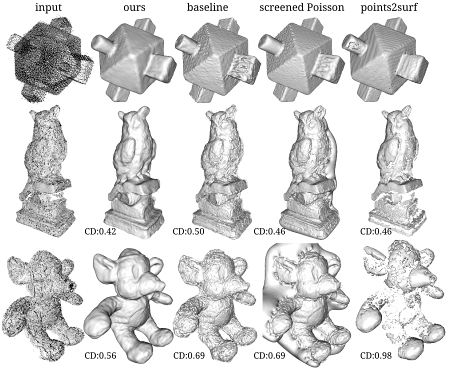
Upsampling



[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

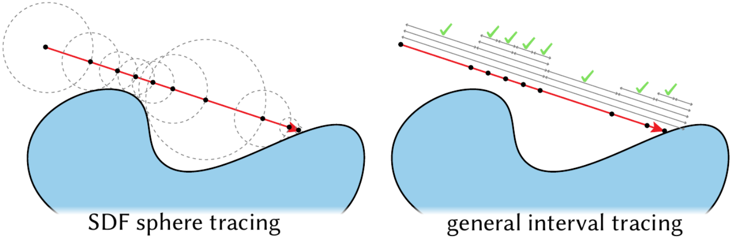
Application to INR fitting regularization



[Yifan 2021]

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

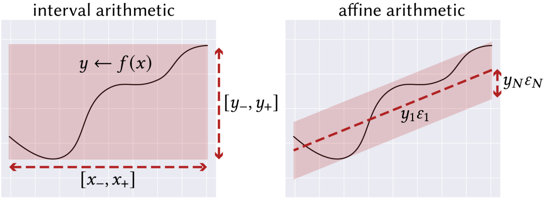
Arithmetic Queries [Sharp 2022]



[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]



[Sharp 2022]

- 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 $\hat{f}(x) \approx \alpha x + \beta$
- $\gamma = \max_{x \in range(\hat{x})} |f(x) - \hat{f}(x)|$
- $\hat{y} = f(\hat{x}) = \alpha x_0 + \beta + \sum_{i=1}^{N} \alpha x_i \varepsilon_i + \gamma \varepsilon_{N+1}$

Nonlinearities

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

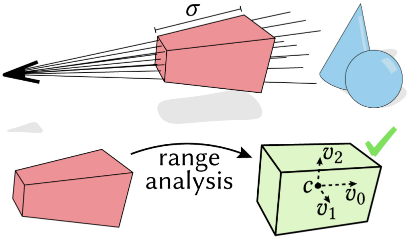
$\hat{\mathbf{y}} \leftarrow f_\theta(\hat{\mathbf{x}})$ ▷ *Propagate affine bounds (Section 3.2)*

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

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

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

else return `UNKNOWN`



[Sharp 2022]

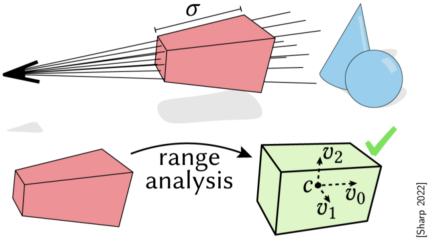
Range bounds

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 $s \le d$ 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`.

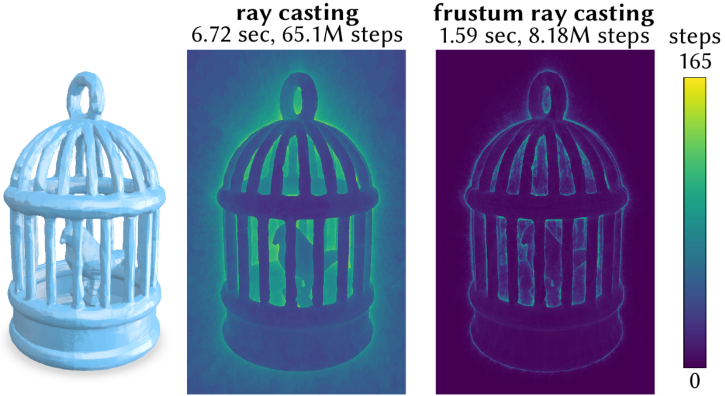
1: $\hat{\mathbf{x}} \leftarrow c + \sum_{i=1}^{s} v_i \varepsilon_i$ ▷ *Construct affine bounds defining the box*
2: $\hat{\mathbf{y}} \leftarrow f_\theta(\hat{\mathbf{x}})$ ▷ *Propagate affine bounds (Section 3.2)*
3: $[y_-, y_+] \leftarrow \mathrm{range}(\hat{\mathbf{y}})$ ▷ *Bound the output (Equation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



Unknown?

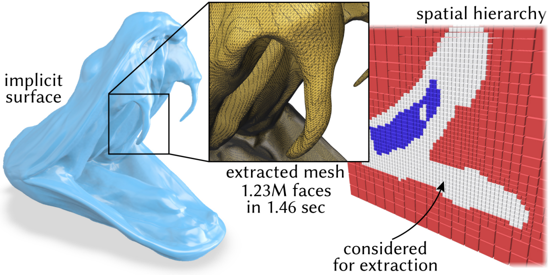
Subdivide the box.

Ray casting vs frustum ray casting

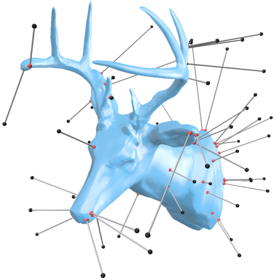


[Sharp 2022]

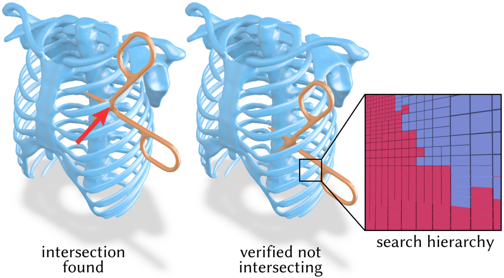
Applications



Mesh extraction



Closest point



Mesh Intersection

Outline

Lipschitz Networks

$$f : \mathcal{X} \to \mathcal{Y}; \forall (x_1, x_2) \in \mathcal{X}^2,\ d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2)$$

Goal

Neural networks are learned functions f_θ from $\mathbb{R}^n$ to $\mathbb{R}^d$, can we design architectures which yield guaranteed K-Lipschitz functions?

Lipschitz Networks

$$f : \mathcal{X} \to \mathcal{Y};\ \forall (x_1, x_2) \in \mathcal{X}^2,\ d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2)$$

Goal

Neural networks are learned functions f_θ from $\mathbb{R}^n$ to $\mathbb{R}^d$, can we design architectures which yield guaranteed K-Lipschitz functions?

With a small K:

- Better generalization

Lipschitz Networks

$$f : \mathcal{X} \to \mathcal{Y};\ \forall (x_1, x_2) \in \mathcal{X}^2,\ d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2)$$

Goal

Neural networks are learned functions f_θ from $\mathbb{R}^n$ to $\mathbb{R}^d$, can we design architectures which yield guaranteed K-Lipschitz functions?

With a small K:

- Better generalization
- Improved adversarial robustness

Lipschitz Networks

$$f : \mathcal{X} \to \mathcal{Y};\ \forall (x_1, x_2) \in \mathcal{X}^2,\ d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2)$$

Goal

Neural networks are learned functions f_θ from $\mathbb{R}^n$ to $\mathbb{R}^d$, can we design architectures which yield guaranteed K-Lipschitz functions?

With a small K:

- Better generalization
- Improved adversarial robustness
- Greater *interpretability*

Lipschitz Networks

$$f : \mathcal{X} \to \mathcal{Y};\ \forall (x_1, x_2) \in \mathcal{X}^2,\ d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2)$$

Goal

Neural networks are learned functions f_θ from $\mathbb{R}^n$ to $\mathbb{R}^d$, can we design architectures which yield guaranteed K-Lipschitz functions?

With a small K:

- Better generalization
- Improved adversarial robustness
- Greater *interpretability*
- Wasserstein distance computation (Peyré & Cuturi 2018).

Lipschitz Networks

$$f : \mathcal{X} \to \mathcal{Y};\ \forall (x_1, x_2) \in \mathcal{X}^2,\ d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2)$$

Goal

Neural networks are learned functions f_θ from $\mathbb{R}^n$ to $\mathbb{R}^d$, can we design architectures which yield guaranteed K-Lipschitz functions?

With a small K:

- Better generalization
- Improved adversarial robustness
- Greater *interpretability*
- Wasserstein distance computation (Peyré & Cuturi 2018).
- Issue: Lipschitz guarantee without sacrificing expressive power.

Notations

- x input, y output
- L layers
- l^{th} layer: dimension n_l, $\mathrm{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$
- $z_l = \mathrm{W}_l h_{l-1} + b_l$, $h_l = \phi(z_l)$
- $y = z_L$
- $C_L(X, \mathbb{R})$ space of all 1-Lipschitz functions mapping (X, d_X) to $(\mathbb{R}, L_p)$

A first result [Anil 2019]

Composition

Composition of two 1-Lipschitz functions is 1-Lipschitz.

A first result [Anil 2019]

Composition

Composition of two 1-Lipschitz functions is 1-Lipschitz.

Consequence

Compose 1-Lipschitz affine transform ($\|Wx\|_p \leq \|x\|_p, \forall x$) and 1 - Lipschitz activations.

A first result [Anil 2019]

Composition

Composition of two 1-Lipschitz functions is 1-Lipschitz.

Consequence

Compose 1-Lipschitz affine transform ($\|Wx\|_p \leq \|x\|_p, \forall x$) and 1 - Lipschitz activations.

- ReLU, tanh, maxout are 1-Lipschitz (if scaled appropriately)!

So... Are we done?

Theorem

Expressivity [Anil 2019] Consider a neural net $f : \mathbb{R}^n \to \mathbb{R}$*, built with* $\|W\|_2 \leq 1$ *and* 1*-Lipschitz elementwise monotonic activation functions. If* $\|\nabla f\|_2 = 1$ *almost everywhere then* f *is linear*

So... Are we done?

Theorem

Expressivity [Anil 2019] Consider a neural net $f : \mathbb{R}^n \to \mathbb{R}$, built with $\|W\|_2 \leq 1$ and 1-Lipschitz elementwise monotonic activation functions. If $\|\nabla f\|_2 = 1$ almost everywhere then f is linear

- ReLU, sigmoid, tanh?

Semi definite Programming Layer [Araujo et al. 2019]

SDPL

Residual layer with parameters $W \in \mathbb{R}^{k \times k}$, $q \in \mathbb{R}^k$, $b \in \mathbb{R}^k$

$$x \leftarrow x - 2WT^{-1}\sigma(W^T x + b)$$

with:

$$T = \sum_{j=1}^{K} |(W^T W)_{ij} \exp(q_i - q_j)|$$

and σ the ReLU activation function.

- W weight matrices are square (0-padding on the input)

Semi definite Programming Layer [Araujo et al. 2019]

SDPL

Residual layer with parameters $W \in \mathbb{R}^{k \times k}$, $q \in \mathbb{R}^k$, $b \in \mathbb{R}^k$

$$x \leftarrow x - 2WT^{-1}\sigma(W^T x + b)$$

with:

$$T = \sum_{j=1}^{K} |(W^T W)_{ij} \exp(q_i - q_j)|$$

and σ the ReLU activation function.

- W weight matrices are square (0-padding on the input)
- Output layer: affine layer

$$x \leftarrow \frac{w^T x}{\|w\|_2} + b$$

Application to signed distance field [Coiffier 2024]

- Set of points x_i with known distances d_i
- Naive approach: combining 1-lipschitz network with a fitting loss:

Application to signed distance field [Coiffier 2024]

- Set of points x_i with known distances d_i
- Naive approach: combining 1-lipschitz network with a fitting loss: underestimates the signed distance field + not robust to imperfect data.

Application to signed distance field [Coiffier 2024]

- Set of points x_i with known distances d_i
- Naive approach: combining 1-lipschitz network with a fitting loss: underestimates the signed distance field + not robust to imperfect data.

Hinge-Kantorovich-Rubinstein loss [Serrurier 2021]

$$\mathcal{L}_{hKR} = \mathcal{L}_{KR} + \lambda \mathcal{L}^m_{hinge}$$

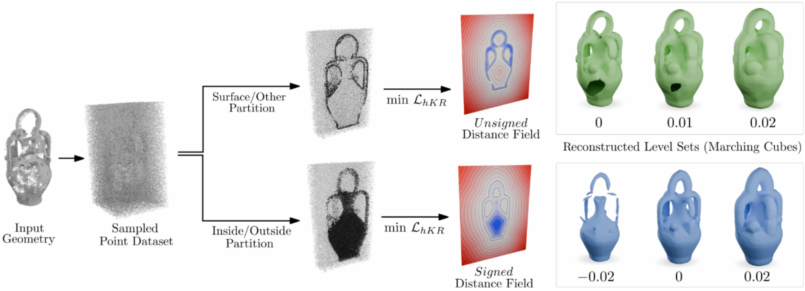
with:

$$\mathcal{L}_{KR} = \sum_i -sign(d_i) u_\theta(x_i)$$

$$\mathcal{L}^m_{hinge} = \sum_i \max(0, m - sign(d_i) u(x_i))$$

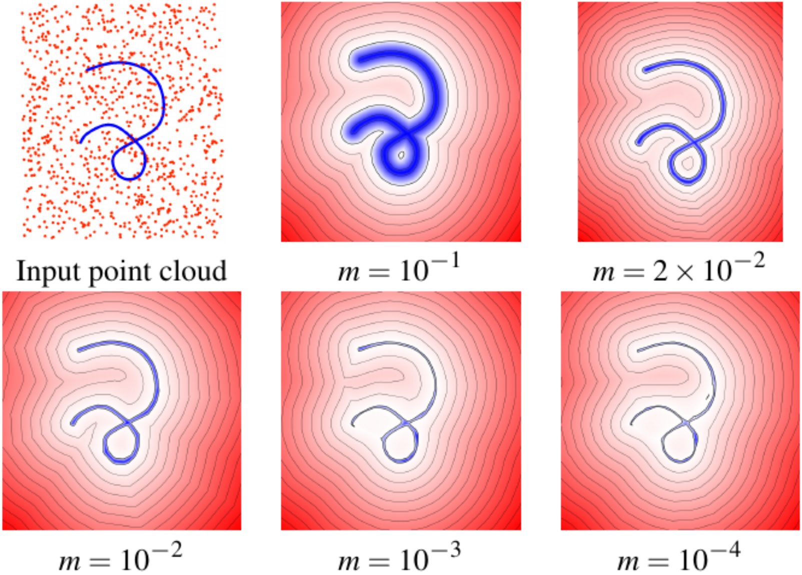
- Under mild assumptions, proof that this converges to an approximation of the signed distance field.

Application to Signed distance field estimation



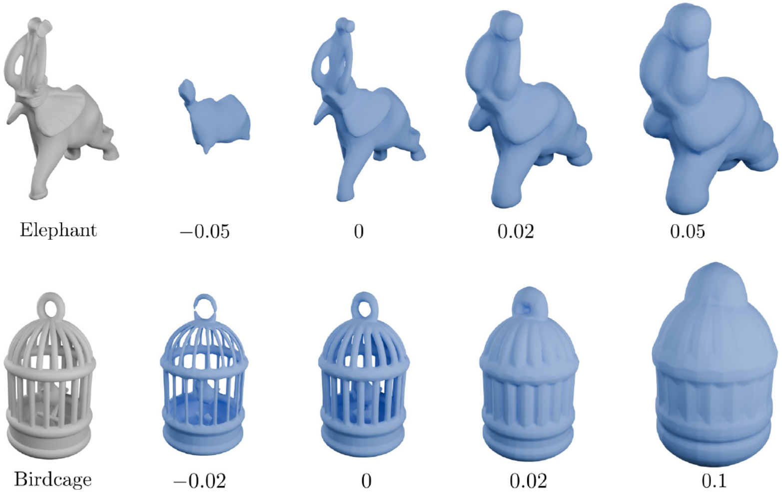
[Coiffier 2024]

Application to Signed distance field estimation



[Coiffier 2024]

Application to Signed distance field estimation



[Coiffier 2024]

Conclusion

- Overview of *Single shape* implicit representation techniques
- Signed distance field or occupancy function or ??
- Combining losses with adequate architectures.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As the
unprocessed data that should have been added to the final page this e
has been added to receive it.
If you rerun the document (without altering it) this surplus page will g
because LaTeX now knows how many pages to expect for this documen