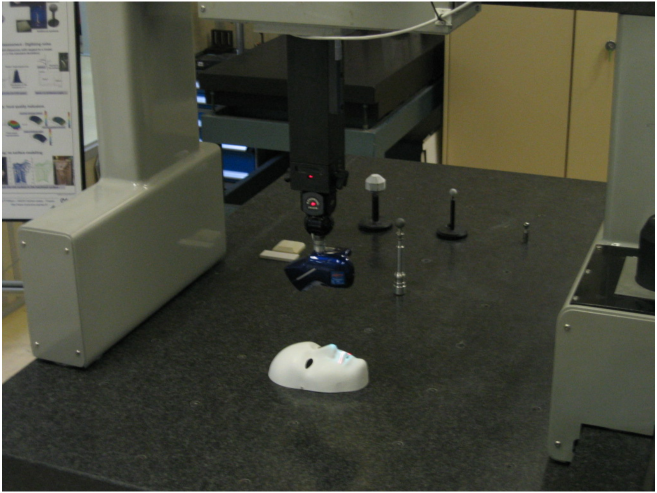
# Reconstruction de formes 3D

Julie Digne

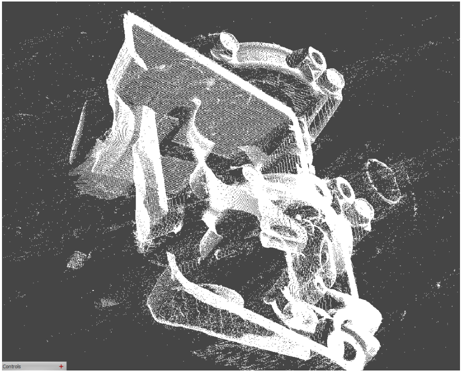
LIRIS - CNRS

20 Octobre 2023

# Introduction: Acquisition of point clouds

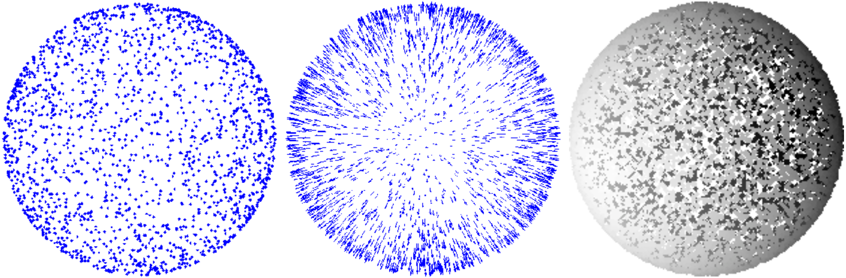


# 3d surfaces typical challenges: Cleaning the physical measure

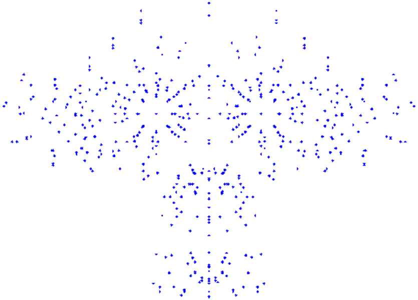


# 3d surfaces typical challenges: Registering and merging scans

# 3d surfaces typical challenges: Orienting the point set

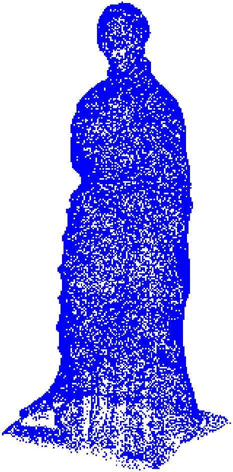


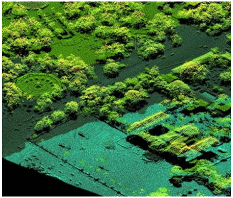
# 3d surfaces typical challenges: Building a mesh from a set of points

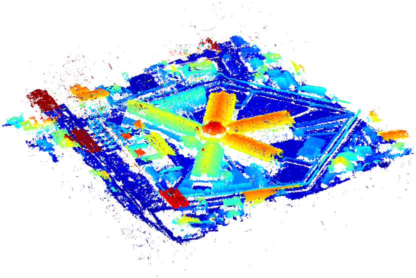


**Shape courtesy of *blender***

# Results of the acquisition process







# Surface reconstruction: for what purpose?

- Interpolating/Approximating?
- Closed surface reconstruction? Boundary preserving surface reconstruction?
- Smooth/piecewise smooth surface?
- Detail preservation/representation sparsity?

Different reconstruction methods depending on the application

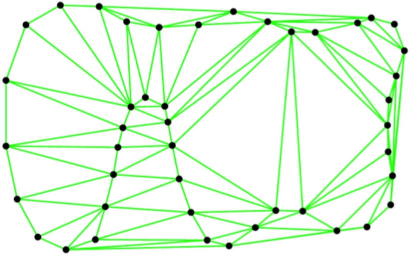
# Outline

# Methods coming from computational geometry

- Convex Hulls...
- Crust, Eigencrust, powercrust
- Delaunay filtering
- $\alpha$-shapes
- Ball Pivoting Algorithm

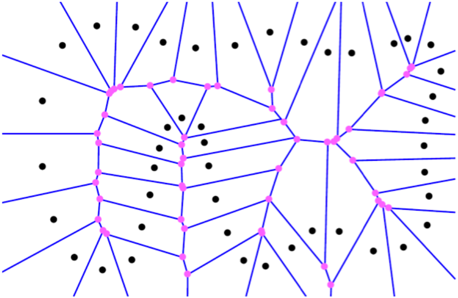
# Computational Geometry

- A Delaunay Triangulation of $S$ is the set of all triangles with vertices in S whose circumscribing circle contains no other points in $S^*$.
- *Compactness Property:* this is a triangulation that maximizes the min angle



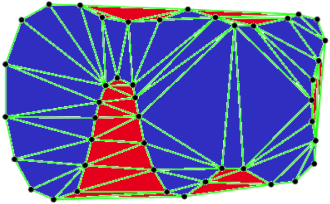
# Computational Geometry

- The Voronoi Diagram of $S$ is a partition of space into regions $V(p)$ ($p \in S$) such that all points in $V(p)$ are closer to $p$ than any other point in $S$.
- For a vertex, we can draw an empty circle that just touches the three points in $S$ around the vertex.
- Each Voronoi vertex is in one-to-one correspondence with a Delaunay triangle



# Space partitioning

- Given a set of points, we can construct the Delaunay triangulation
- Label each triangle/tetrahedron as inside/outside
- Reconstruction = set of edges/facets that lie between inside and outside triangles/tetrahedra
- Different ways of assigning the labels [Boissonat 84], tight cocoone [Dey Goswami 2003], Powercrust [Amenta et al. 2001] Eigencrust [Kolluri et al. 2004]



# The Crust Algorithm [Amenta et al. 1998]

- If we consider the Delaunay Triangulation of a point set, the shape boundary can be described as a subset of the Delaunay edges.
- How do we determine which edges to keep?
- Two types of edges:
  - Those connecting adjacent points on the boundary
  - Those traversing the shape
- Discard those that traverse the shape

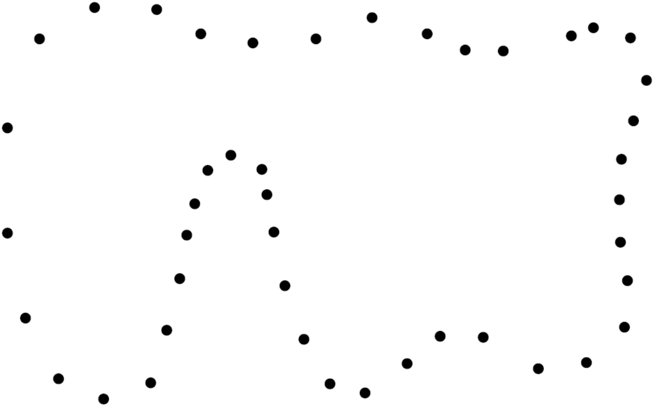
# The Crust Algorithm [Amenta et al. 1998]

In 2D:

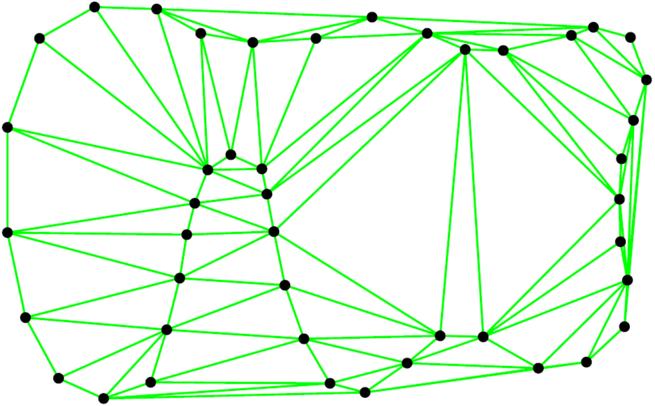
- Given a point set $S$ compute its Voronoi diagram and Voronoi vertices $V$
- Compute the Delaunay triangulation of $S \cup V$
- Keep only edges that connects points in $S$ (eq. to Keeping all edges for which there is a circle that contains the edge but no Voronoi vertices)

In 3D: Not all Voronoi Vertices are added to the set. Only the poles (furthest points of the Voronoi cell) are considered.

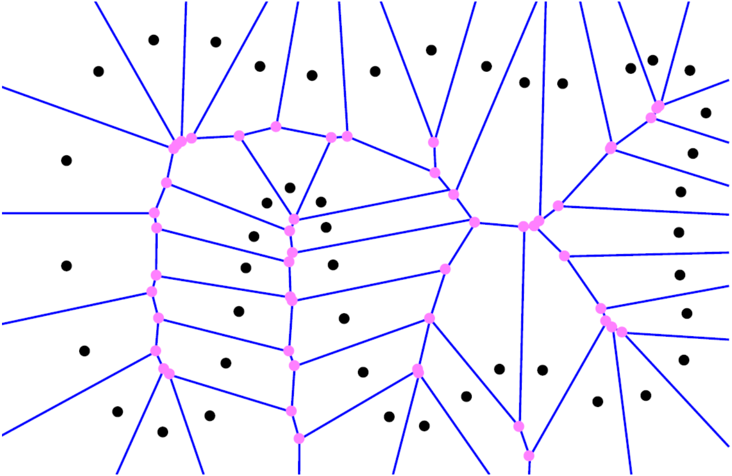
# Crust



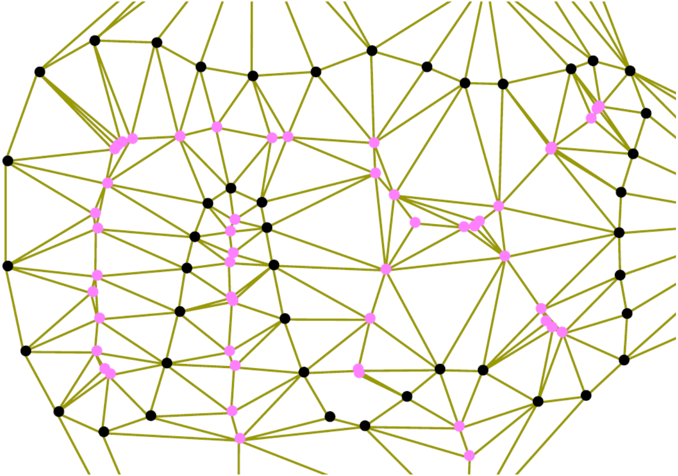
# Crust



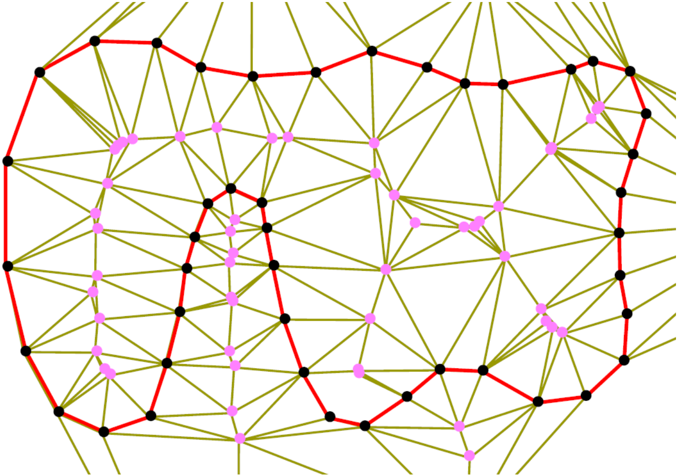
# Crust



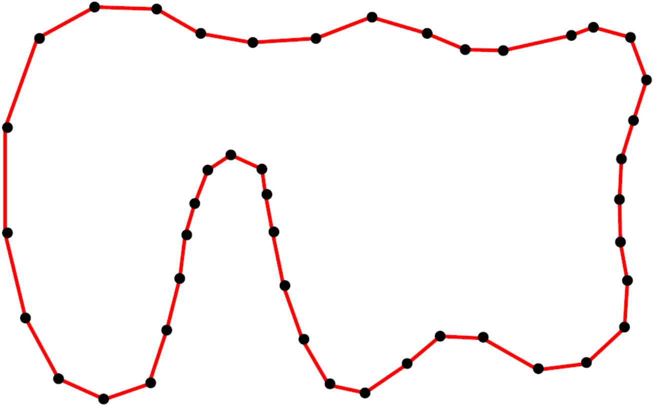
# Crust



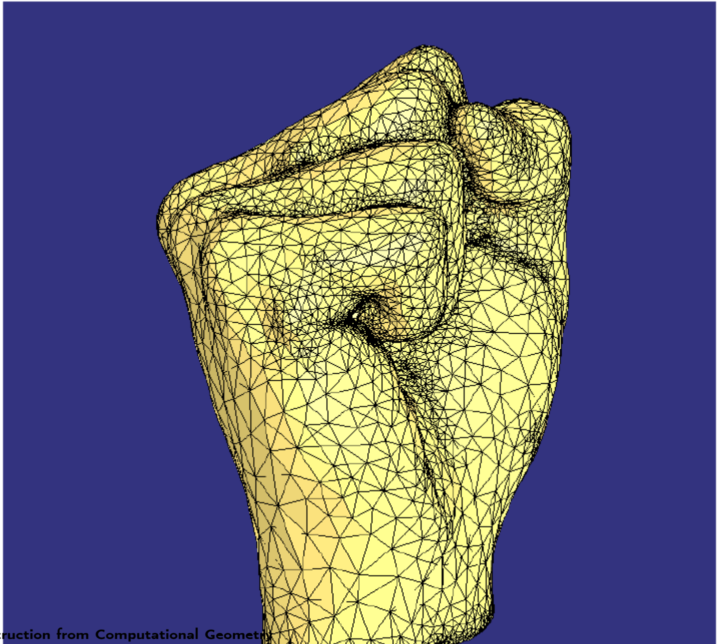
# Crust



# Crust



# Crust Result



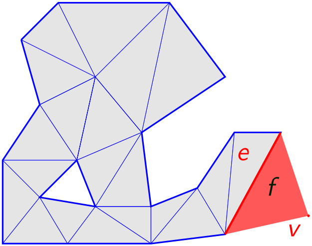
# Ball Pivoting Algorithm

- BPA computes a triangle mesh *interpolating* a given point cloud
- Three points form a triangle if a ball of a user-specified radius $\rho$ touches them without containing any other point
- Start with a seed triangle
- The ball pivots around an edge until it touches another point, forming another triangle
- Expand the triangulation over all edges then start with a new seed

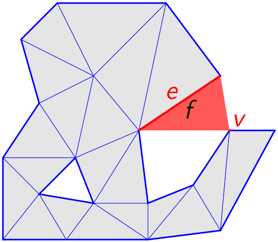
# Algorithm

- Advancing front triangulation
- Front is a set of edges

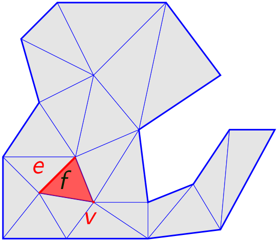
# Different types of expansion



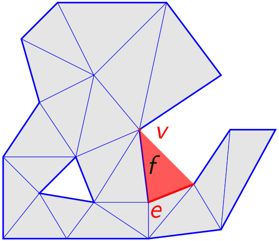

(f) Expansion case




(g) Gluing case

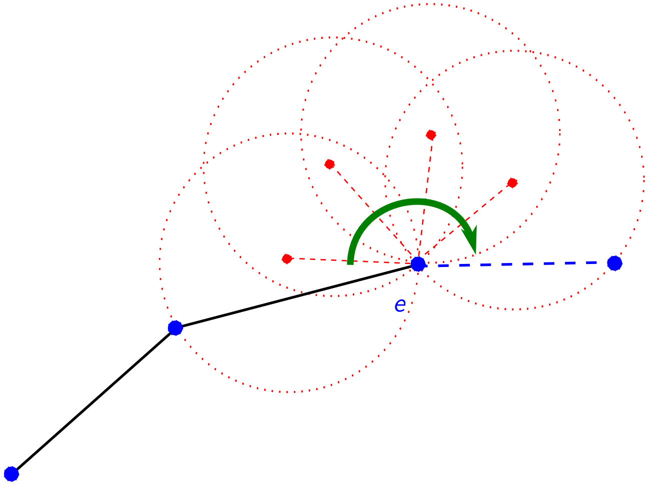



(h) Hole filling case

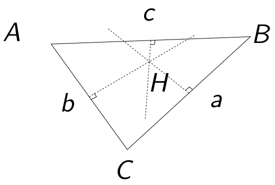


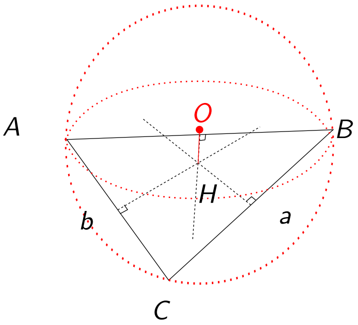

(i) Ear filling case

# Rotating the sphere



# Finding the $R$-circumsphere



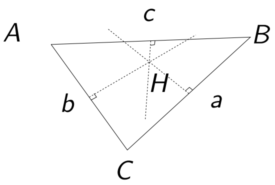

- The circumcenter $H$ of triangle $ABC$ has barycentric coordinates:

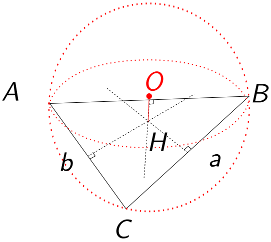
$$\left(a^2(b^2+c^2-a^2), b^2(a^2+c^2-b^2), c^2(a^2+b^2-c^2)\right)$$

- the square circumradius is

$$R^2 = \frac{a^2 \cdot b^2 \cdot c^2}{(a+b+c)\cdot(b+c-a)\cdot(c+a-b)\cdot(a+b-c)}.$$

# Finding the $R$-circumsphere



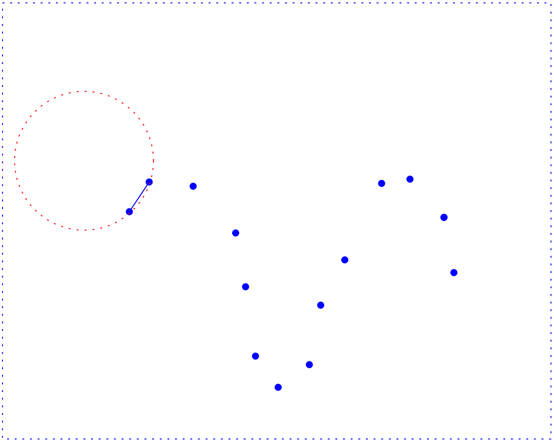

- Such a sphere exists only if $R_b^2 - R^2 \geq 0$.
- Let us denote by n the normal to the triangle plane, oriented such that is has a nonnegative scalar product with the vertices normals. Provided $R_b^2 - R^2 \geq 0$ (hence the sphere existence), the center $O$ of the sphere can be found as:

$$O = H + \sqrt{R_b^2 - R^2} \cdot \mathrm{n}.$$

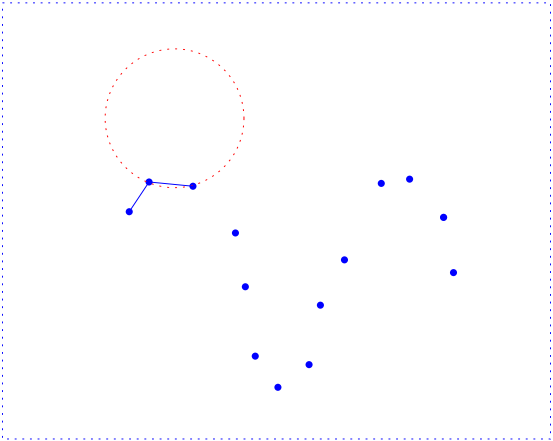
# Properties and Guarantees of the resulting mesh

- The surface is guaranteed to be self-intersection free (no triangle will intersect each other except at an edge or vertex, and at most two triangles can be adjacent to an edge).
- Normal coherence on a facet.
- For each triangle there exists an empty ball incident to the three vertices with empty interior

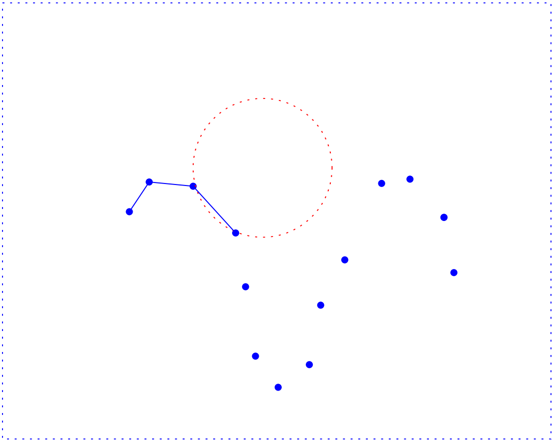
# Detailed area



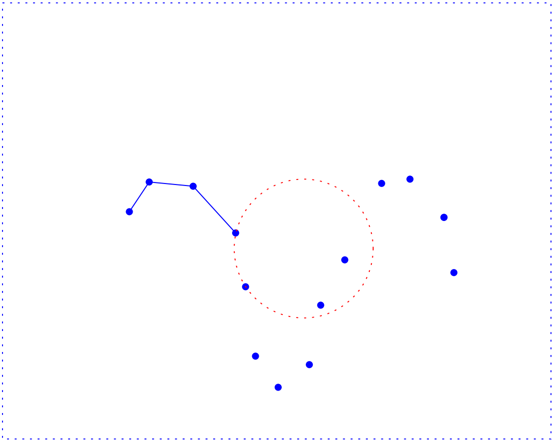
# Detailed area



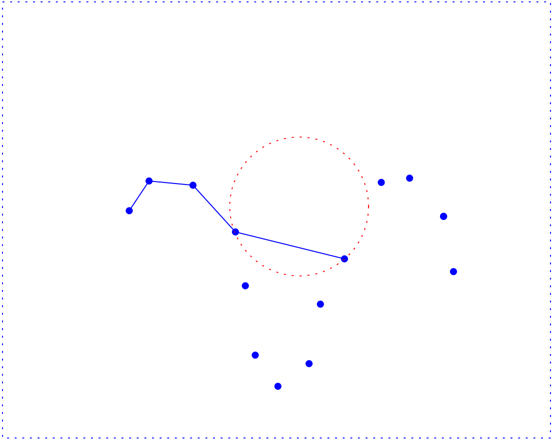
# Detailed area



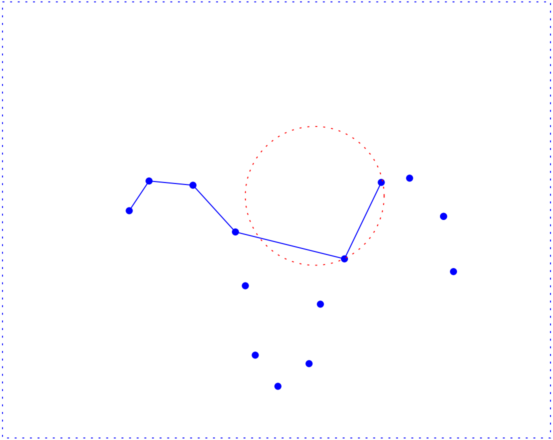
# Detailed area



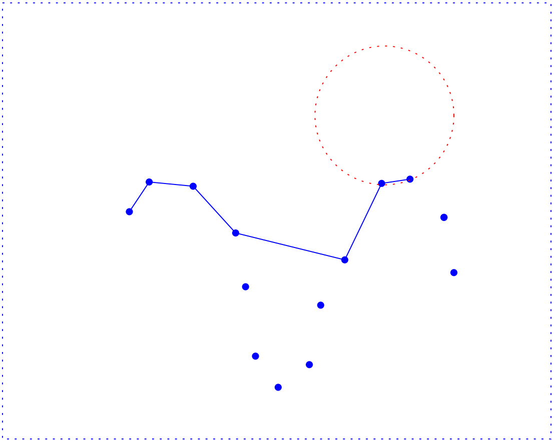
# Detailed area



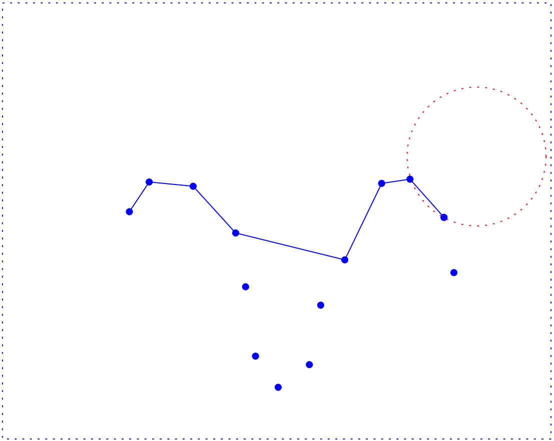
# Detailed area



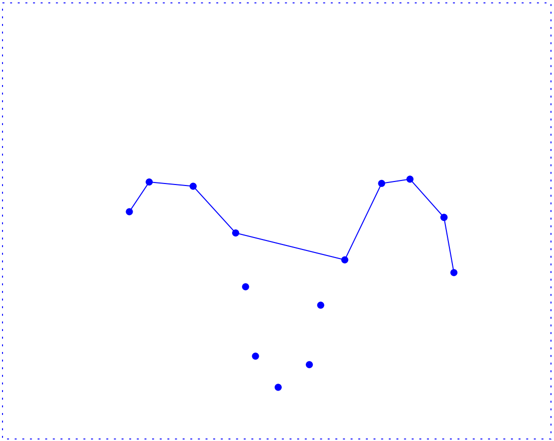
# Detailed area



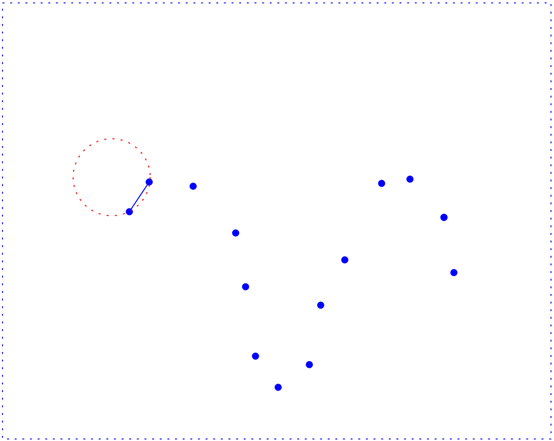
# Detailed area



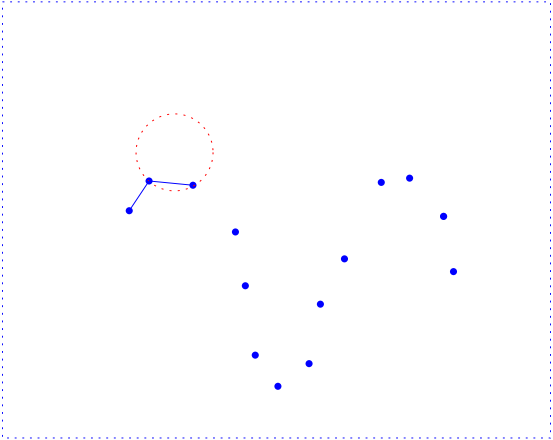
# Detailed area



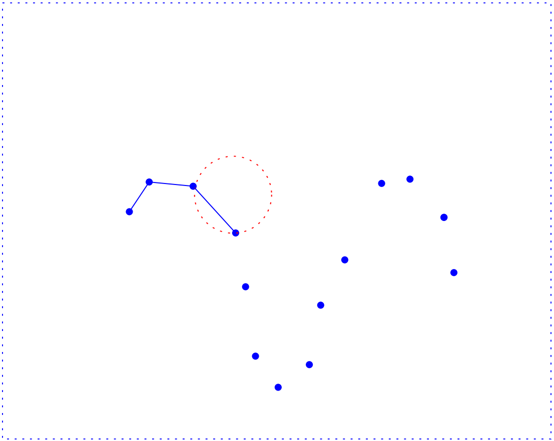
# Smaller ball radius



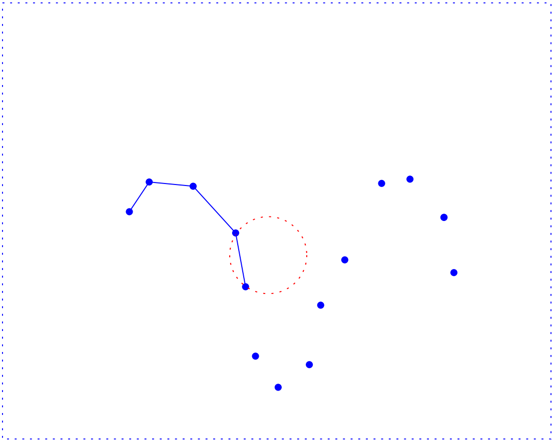
# Smaller ball radius



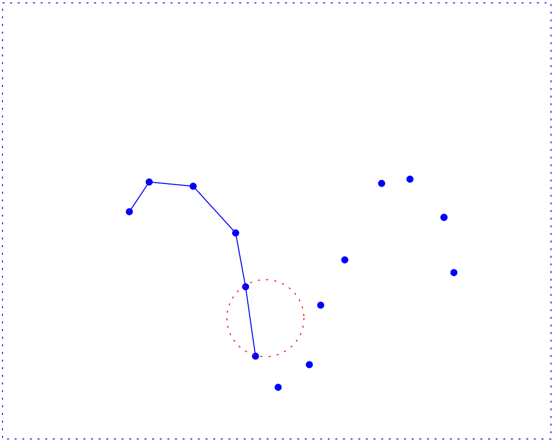
# Smaller ball radius



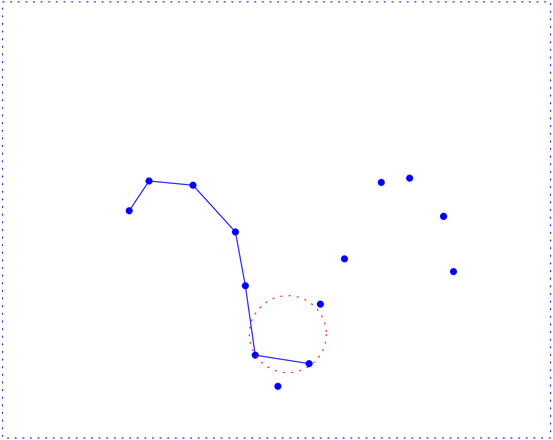
# Smaller ball radius



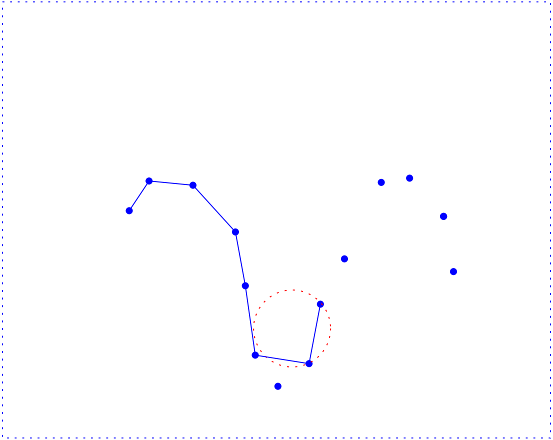
# Smaller ball radius



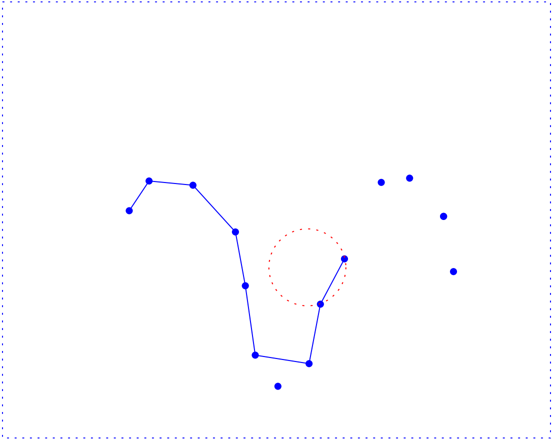
# Smaller ball radius



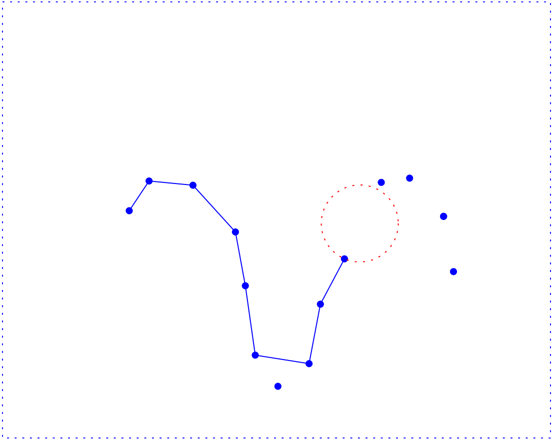
# Smaller ball radius



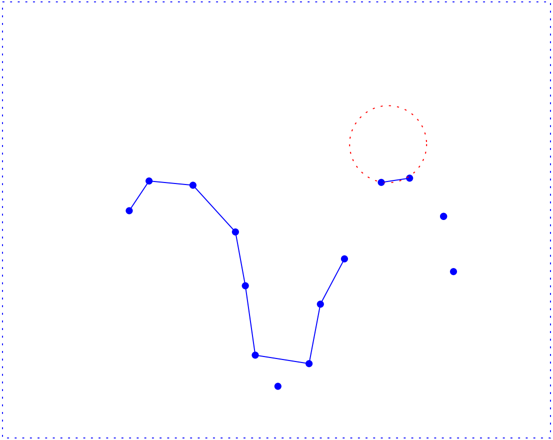
# Smaller ball radius



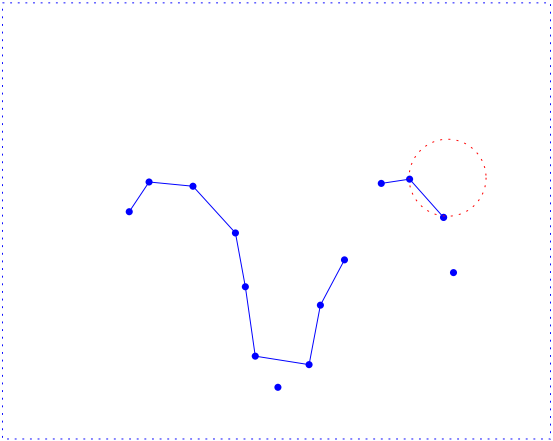
# Smaller ball radius



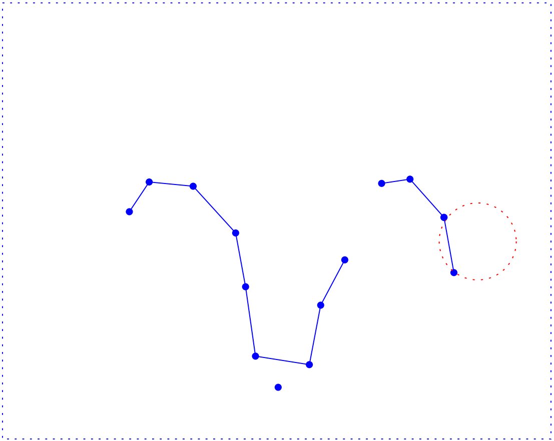
# Smaller ball radius

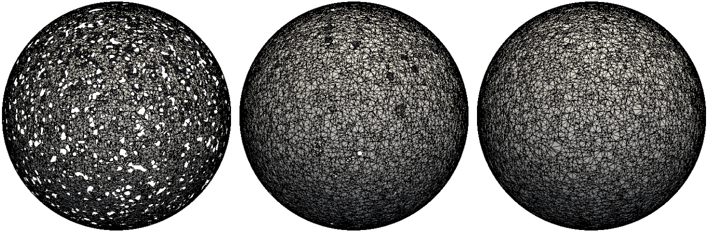


# Smaller ball radius



# Smaller ball radius



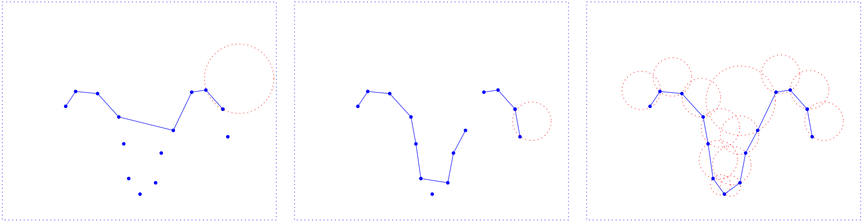


(j) $r = 0.02$ (k) $r = 0.03$ (l) $r = 0.05$

Figure: Radius too small: areas with lower density are not triangulated. Large radius : higher computation times + detail loss.

Figure: Reconstructing the Stanford Bunny point cloud, with a single radius (0.0003), two radii (0.0003; 0.0005) and three radii (0.0003; 0.0005; 0.002).

| Radius | Time(s) | vertices | facets | boundary edges |
|---|---|---|---|---|
| 0.0003 | $10s$ | 318032 | 391898 | 272832 |
| 0.0003; 0.0005 | $21s$ | 356252 | 698963 | 22727 |
| 0.0003; 0.0005; 0.002 | $29s$ | 361443 | 713892 | 7897 |

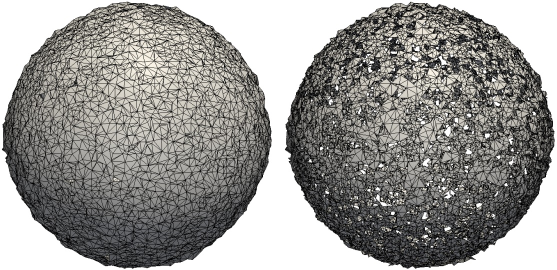


(a) Detail loss

(b) Hole creation

(c) A possible correction: using multiple radii

Figure: Detail loss and hole creation due to a too large radius (left) and a too small one (middle). A possible solution is to use multiple radii (right).



(a) $r = 0.05$ (b) $r = 0.02; 0.03; 0.05$

Figure: Applying the ball pivoting to a noisy sphere: $r = 0.05$ (left) and $r = 0.02; 0.03; 0.05$ (right). A single radius does not allow to interpolate the input data and applying multiple radii is not a solution in addition to being difficult to tune.

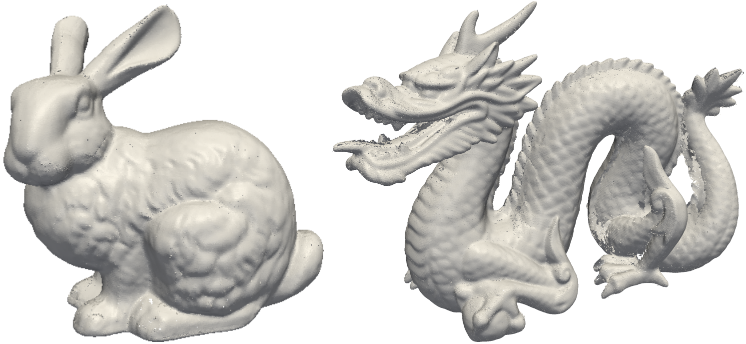


Figure: Bunny and Dragon reconstruction

# Problems and solutions

- The larger the ball radius the slower the computation
- The larger the ball radius the more details will be lost
- The smaller the ball radius the more dependent on the sampling
- Varying ball radius $\leftarrow$ slow down the process
- Use of a *scale space*: a multiscale representation of the point cloud.

# Summary: Advantages/Drawbacks of the ball pivoting

Drawbacks

- Size of the ball?
- No suppression of redundant points
- No hole closure

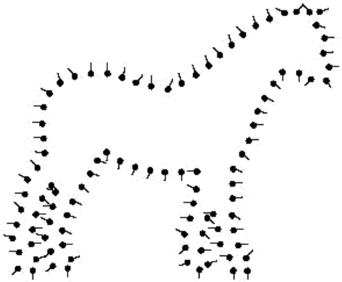
Advantages

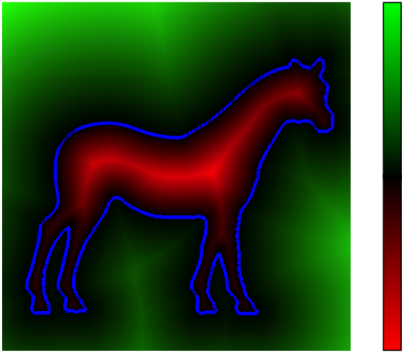
- Control on the size of the triangles created
- Radius of the ball determines what is a hole
- Surface boundary preservation

Modification through the use of a *scale space* for better detail preservation.

# Outline

# Implicit surface reconstruction - Principle





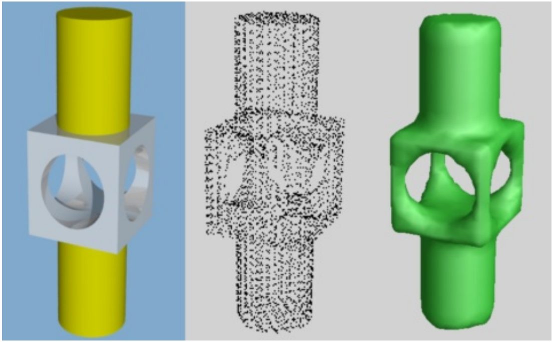
- See the surface as an isolevel of a given function
- Extract the surface by some contouring algorithm: Marching cubes [Lorensen Cline 87], Particle Systems [Levet et al. 06]

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

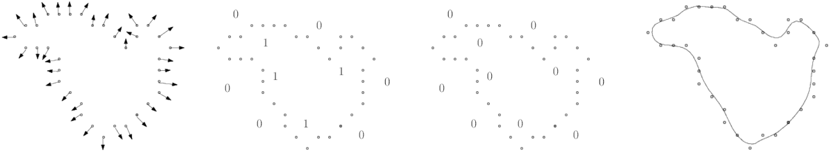
- Input: a set of 3D points
- *Idea:* for points on the surface the signed distance transform has a gradient equal to the normal
$$F(p) = \pm \min_{q \in \mathcal{S}} \|p - q\|$$
- 0 is a regular value for $F$ and thus the isolevel extraction will give a manifold
- Compute an associated tangent plane $(o_i, n_i)$ 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: $\hat{f}(p) = n_i \cdot (p - o_i)$



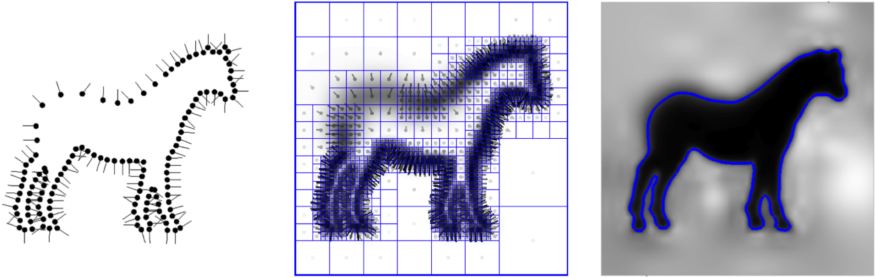
# Poisson Surface Reconstruction [Kazhdan et al. 2006]



- 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



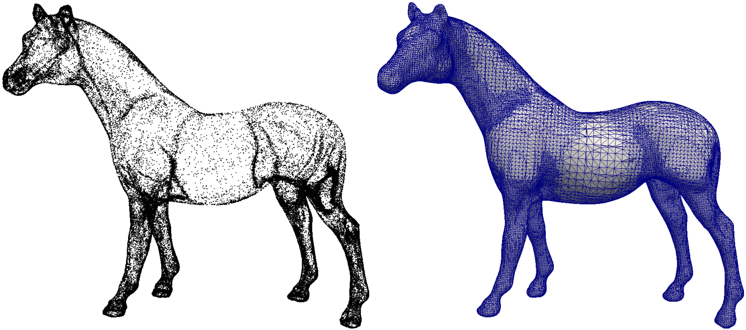
# Poisson Surface Reconstruction [Kazhdan et al. 2006]

- To fit a scalar field $\chi$ 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:

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

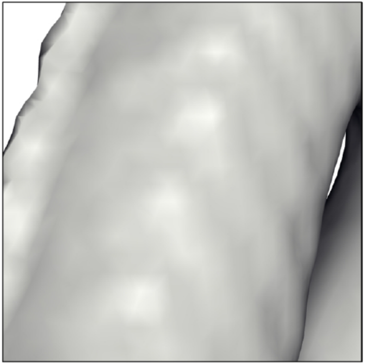
- Let us define $V(q_0) = \sum_s |\mathcal{P}(s)| \tilde{F}(q_0 - s) \cdot \vec{N}(s)$

# Problem Discretization

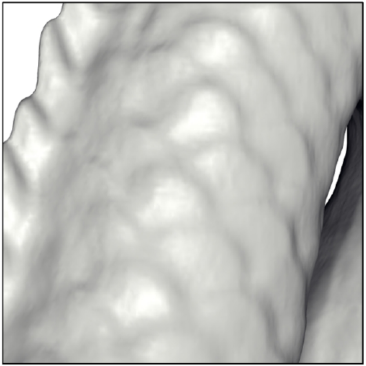
- Build an adaptive octree $\mathcal{O}$
- Associate a function $F_o$ to each node $o$ of $\mathcal{O}$ so that: $F_o(q) = F(\frac{q - o.c}{o.w})\frac{1}{o.w^3}$ ($o.c$ and $o.w$ are the center and width of node $o$). $\Rightarrow$ multiresolution structure
- The base function $F$ is the *nth* convolution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

# Varying octree depth

# Varying octree depth



# Varying octree depth



# Resilience to bad normals

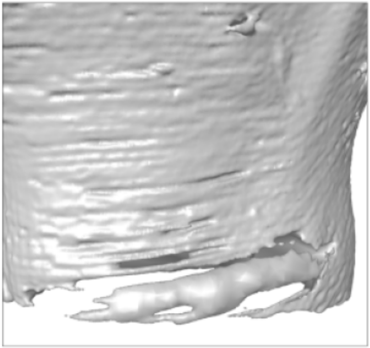
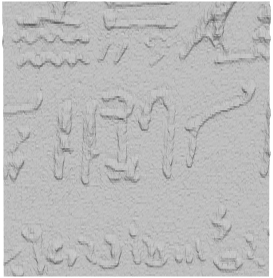


**Image from Mullen et al. Signing the unsigned, 2010**

# detail preservation

Poisson

BPA



Scale Space + BPA

# Moving Least Squares surfaces

**definition**

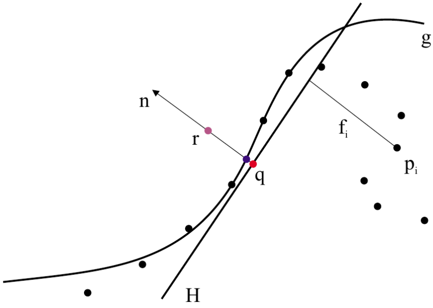
A set of points $(x_i) \in \mathbb{R}^3$ with associated function values $f_i$, Moving least squares approximation

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

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

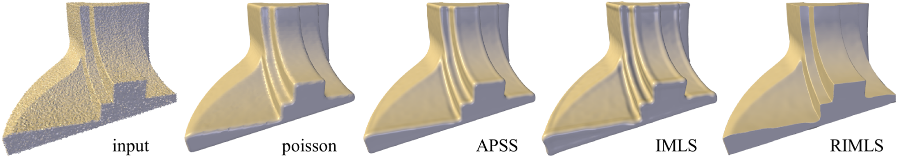
# Adaptation to 3D surfaces

- 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]

# 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

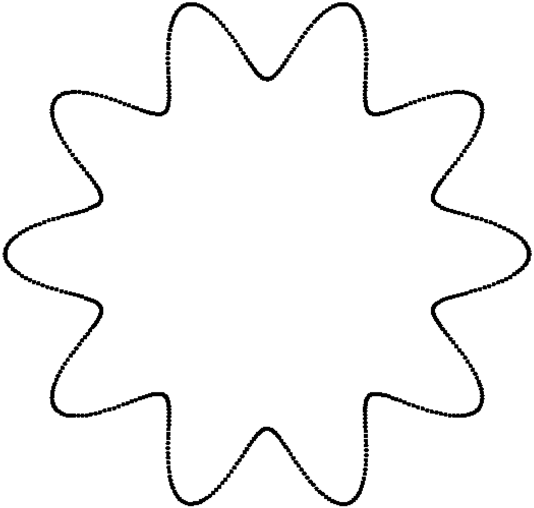
# From the signed distance function to the mesh

- 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

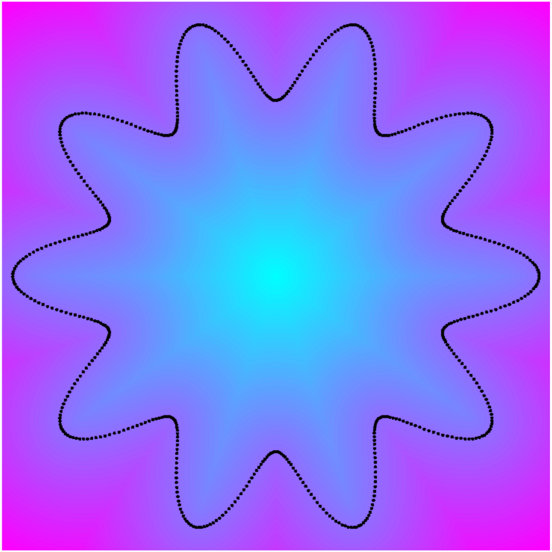
### Approximation

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

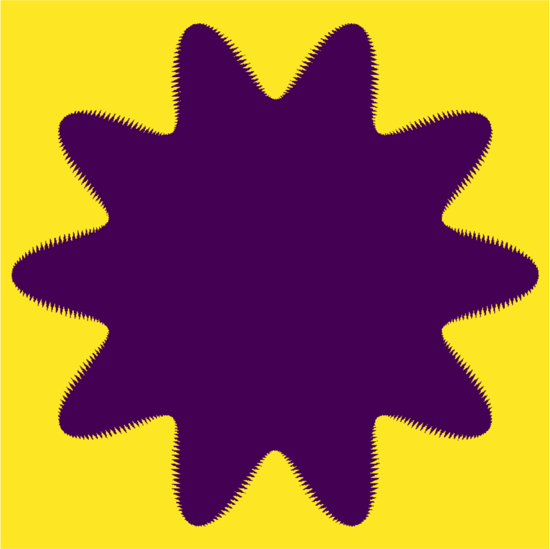
# Example in 2D



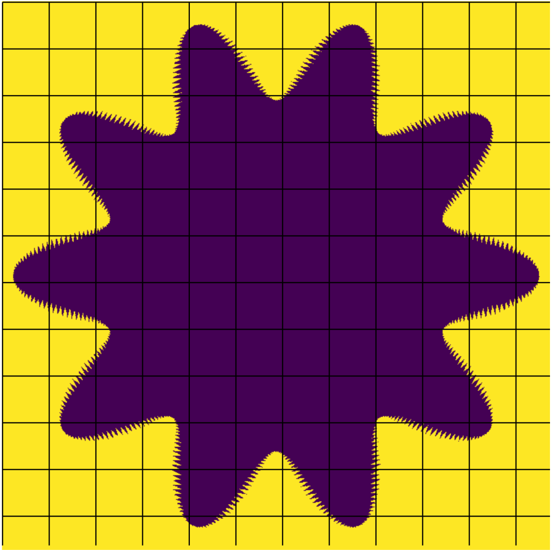
# Example in 2D



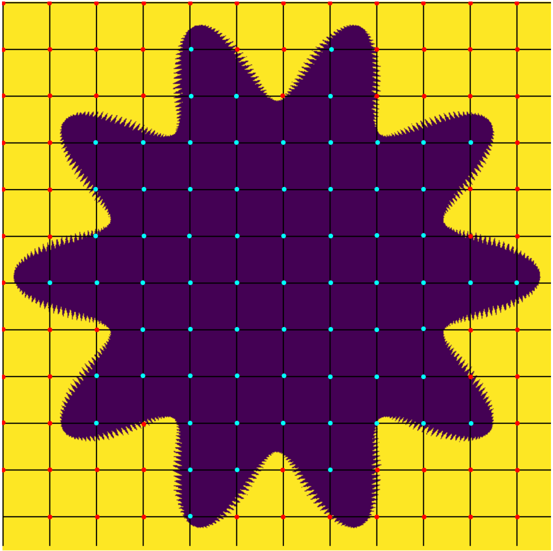
# Example in 2D



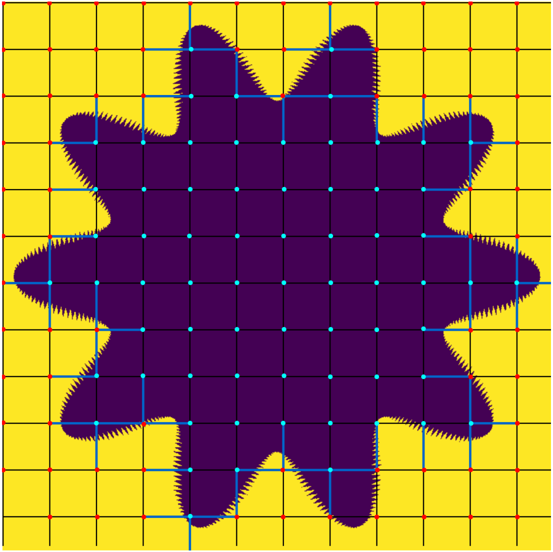
# Example in 2D



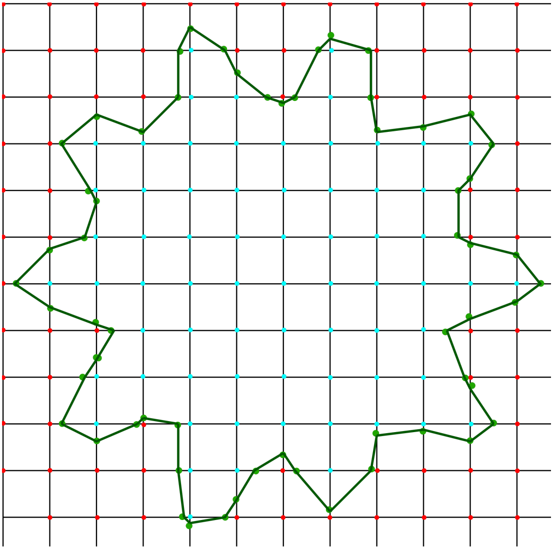
# Example in 2D



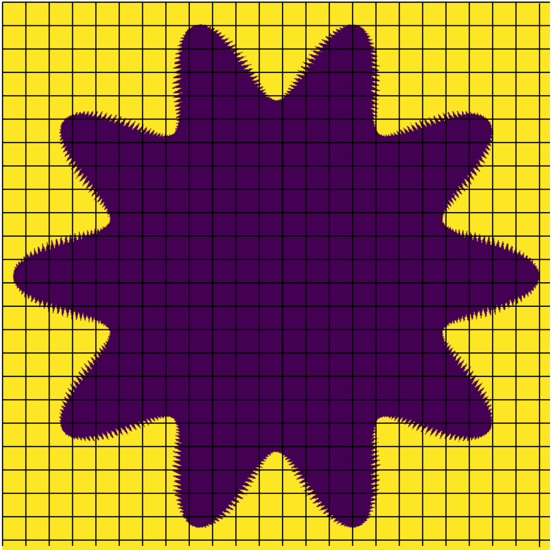
# Example in 2D



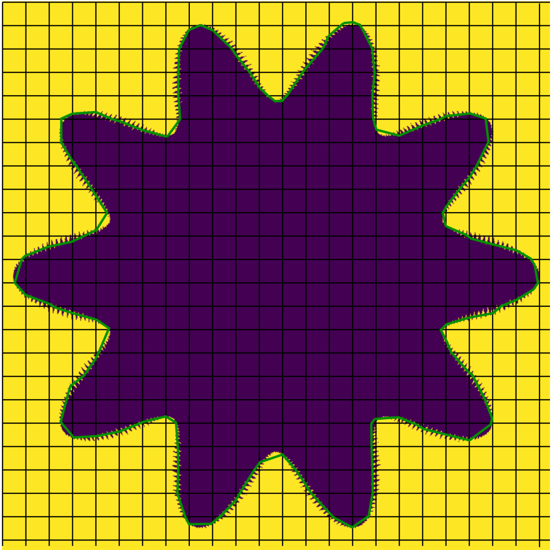
# Example in 2D



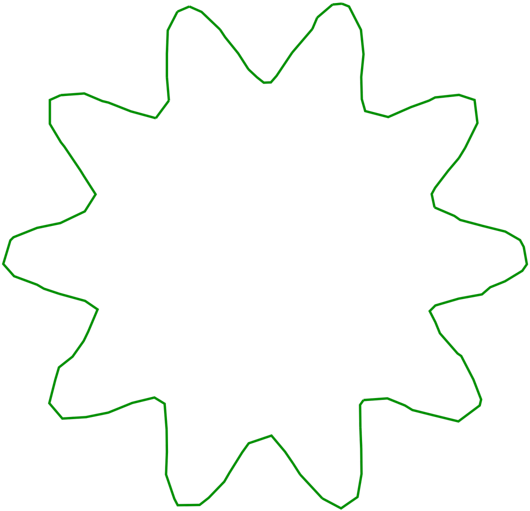
# Example in 2D



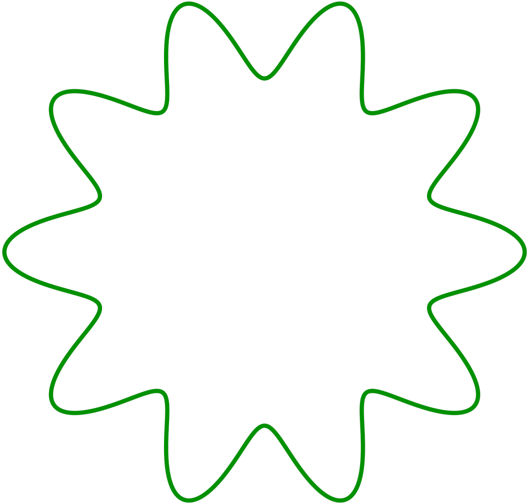
# Example in 2D



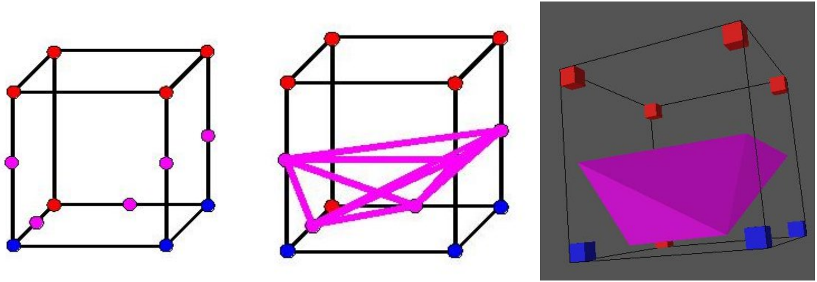
# Example in 2D



# Example in 2D

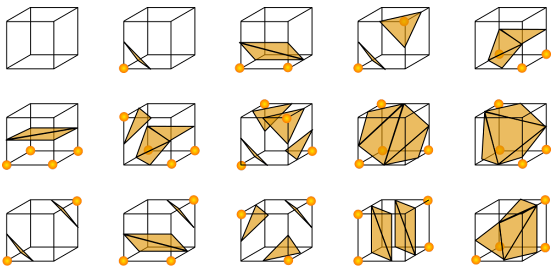


# From Marching Squares to Marching Cubes



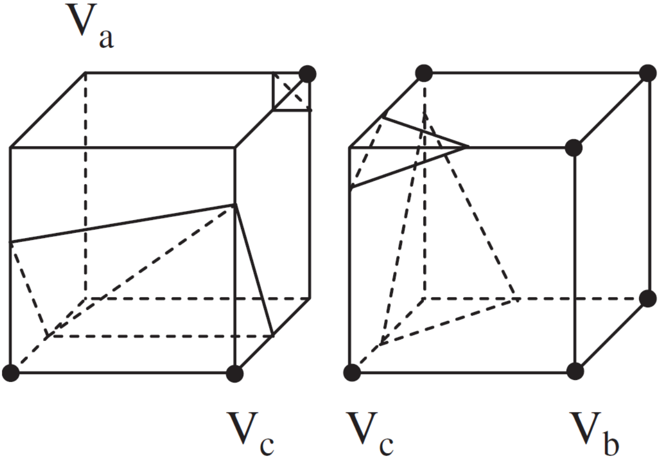
Drawing lines between intersection points is ambiguous and does not give a surface patch.
**Images by Ben Anderson**

# Look-up tables

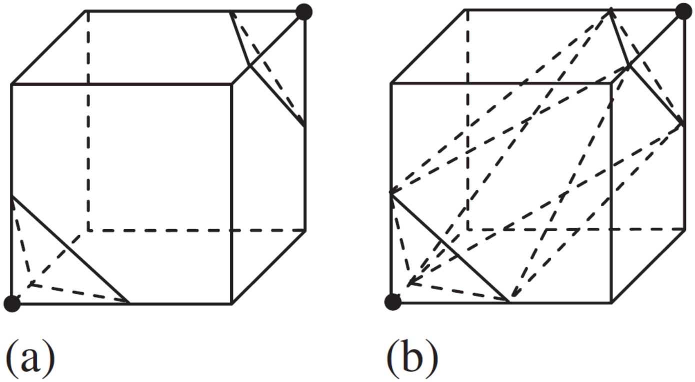


- There are $2^8 = 256$ possible cases for cube corner values.
- By symmetry + rotation arguments it reduces to 15 cases.
- It is thus possible to build a look-up table giving the grid cell triangulation based on the corner values case.

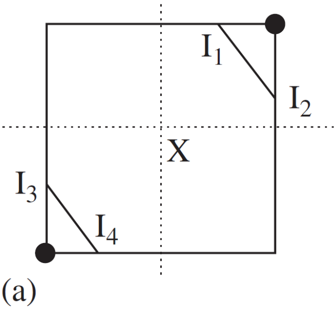
# Ambiguous cases



# Ambiguous cases



# Ambiguous cases



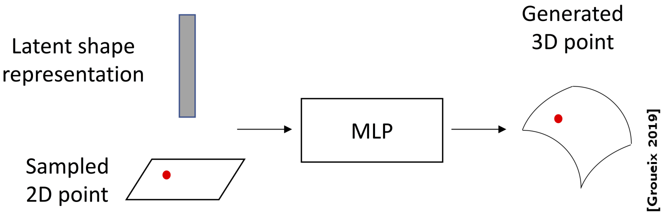

- Refine the grid to remove ambiguation
- Switch to marching tetrahedra algorithm

# Outline

# Machine learning based surface reconstruction

- Needs a differentiable pipeline
- Challenge: intrinsically a combinatorial problem...
- Not necessarily example-based: surface reconstruction can be done per shape.

# AtlasNet [Groueix 2019]




- Some definitions:
  - A manifold surface $\mathcal{S}$ in $\mathbb{R}^3$ is topological set such that each point has a neighborhood which is homeomorphic to an open disk of $mathbbR^2$.
  - Local map (or chart):s a homeomorphism $\varphi$ from an open subset $U$ of $\mathcal{S}$ to an open subset of $\mathbb{R}^2$.
  - Atlas:a indexed family of local charts $(U_i, \phi_i)$ from $U_i$ to open subsets of $\mathbb{R}^2$; such that the $U_i$s cover $\mathcal{S}$.

## Parameterization

This is the base for surface parameterization problems in geometry processing: Try to unwrap a surface onto a planar patch (usually a square).

# AtlasNet [Groueix 2019]

- Model the local maps as affine maps, they can be inverted if they are full rank.
- A ReLU-based MLP computes a piecewise affine map (full rank). This is due to ReLU activation.
- Start with N patches and compute their deformation onto the surface (*Papier mâché*). Deformed patches may overlap.

# AtlasNet for surface reconstruction

- Start with a latent representation $x$ of a shape
- For a set of points $\mathcal{A}$ of points sampled in $[0,1]^2$, we optimize the weights $\theta_i$ of $N$ functions (MLP) $f_{\theta_i}$
- Sample a set $\mathcal{S}_d$ of $M$ points on the surface $\mathcal{S}$
- Chamfer Loss

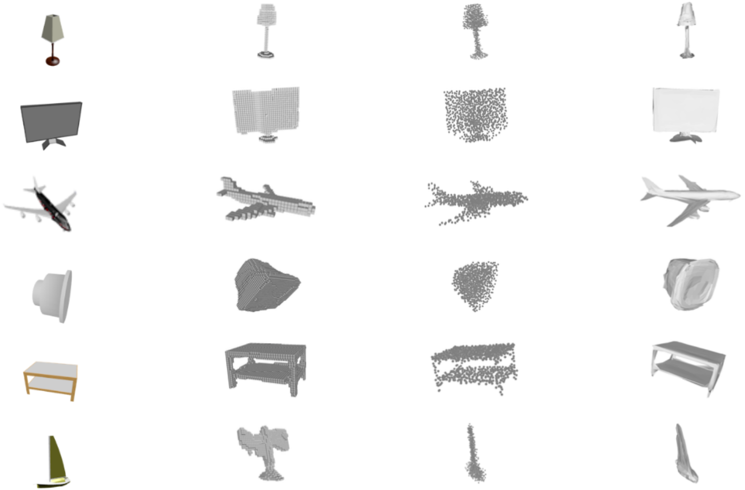
$$\sum_{p\in\mathcal{A}}\sum_{i=1}^{N}\min_{q\in\mathcal{S}_D}\|f_{\theta_i}(p,x)-q\|_2^2+\sum_{q\in\mathcal{S}_d}\min_{i=1\cdots N}\min_{p\in\mathcal{A}}\|f_{\theta_i}(p,x)-q\|^2$$

# Result


(a) Possible Inputs (b) Output Mesh from the 2D Image (c) Output Atlas (optimized) (d) Textured Output (e) 3D Printed Output

[Groueix et al. 2019]

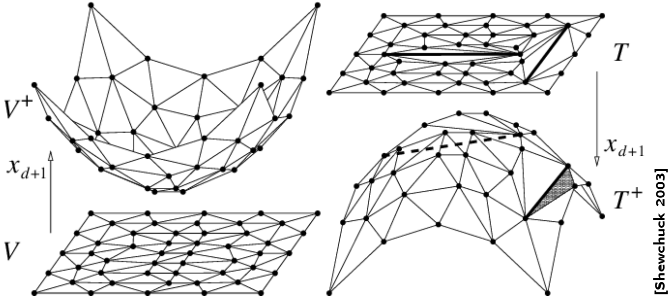
# Results: reconstruction from single view



(a) input (b) 3D-R2N2 (c) PSG (d) Ours

[Groueix et al. 2019]

# Differentiable Surface Reconstruction [Rakotosaona 2021]




- A set of points $v_j \in \mathbb{R}^d$ with weights $w_j$
- *Weighted Delaunay Triangulation:* projected lower envelop of points $(v_j, \|v_j\|^2 - w_j) \in \mathbb{R}^{d+1}$
- Any 2D ($d = 2$) triangulation can be obtained as a perturbation of a 2d Weighted Delaunay Triangulation.

# *Differentiable* ~~weighted~~ Delaunay triangulation in 2D

- All possible triangles with vertices in $V$ are given an inclusion score $e_i$.
- defs: $c_i$ circumcenter of triangle $i = \{j, k, l\}$, $a_{i|j}$ reduced Voronoi cell of vertex $j$ onto triangle $i$. Then

$$e_i = \begin{cases} 1 & \text{if } c_i \in a_{x|i} \ \forall x \in \{j, k, l\} \\ 0 & \text{otherwise} \end{cases}$$

- *Continuous inclusion score*

$$s_{i|j} = \sigma(\alpha d(c_i, a_{j|i})) \ (\sigma \text{ sigmoid})$$

$$s_i = \frac{1}{3}(s_{i|j} + s_{i|k} + s_{i|l})$$

# *Differentiable* weighted Delaunay triangulation in 2D

## Weighted Voronoi cell $a^w$

Intersection of half planes $H_{j\leq k} = \{x \in \mathbb{R}^2 | \|x - v_j\|^2 - w_j \leq \|x - v_k\|^2 - w_k\}$

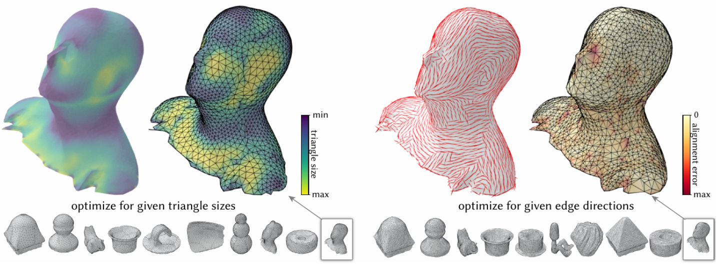
- redefine: $c_i$ weighted circumcenter of triangle $i = \{j, k, l\}$, $a_{i|j}$ reduced weighted Voronoi cell of vertex $j$ onto triangle $i$.
- Same expression for the continuous inclusion score

$$s_{i|j} = \sigma(\alpha d(c_i, a^w_{j|i})) \text{ } (\sigma \text{ sigmoid})$$

$$s_i = \frac{1}{3}(s_{i|j} + s_{i|k} + s_{i|l})$$

# Turning 3D triangulation problems into 2d triangulation problems

- Segment 3D shapes into *developable sets* by Least Squares Conformal Maps [Lévy 2008].
- Differentiable 2D meshing on each of the sets with boundary constraints.




from [Rakotosaona 2021]

# Losses

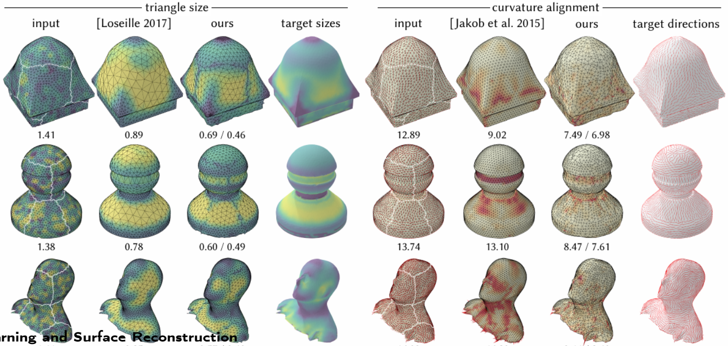
- Area prescribing loss ($A$: function on the surface):

$$\mathcal{L}_{area} = \frac{1}{\sum_{i,j} s_{i|j}} \sum_{i,j} \left(\frac{1}{2} \|(v_j - v_k) \times (v_l - v_k)\| - A(v_j)\right)$$

- Boundary preservation loss:

$$\mathcal{L}_b(V, \mathcal{P}) = \frac{1}{|V|} \sum_j \exp(\varepsilon - \min(\varepsilon, (v_j - b_j) \cdot n_j^b))$$

- Other possible losses: angle loss, curvature alignment loss.

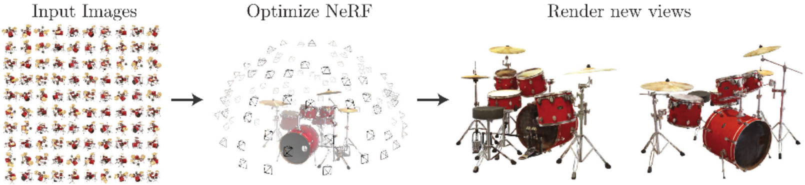



om [Rakotosaona 2021]

# The implicit alternative

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

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



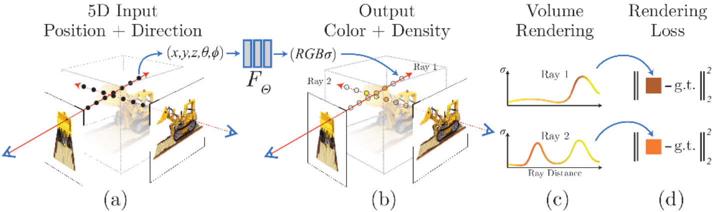

- Goal: Generate a new view from a set of views

### 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)

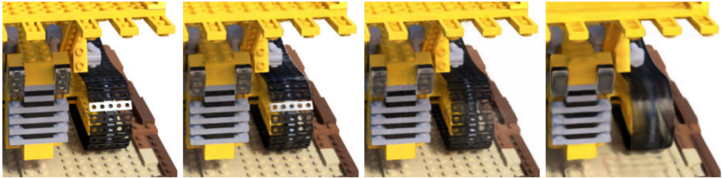
# Training




- 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

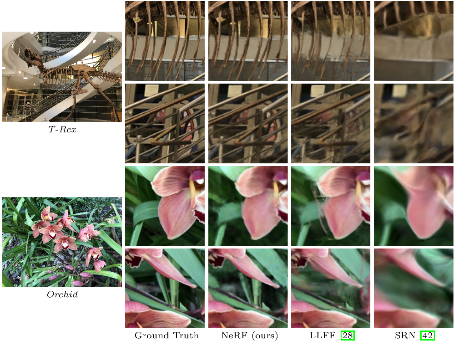
# More tricks

- 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

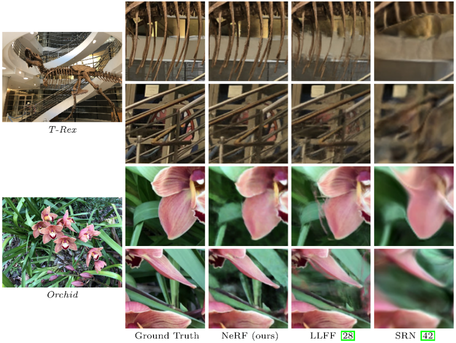



Video: `https://www.matthewtancik.com/nerf`

## 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])*

# Results

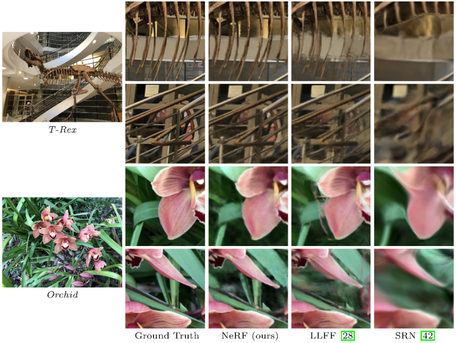



Video: `https://www.matthewtancik.com/nerf`

## 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])*

# 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])*

# Implicit neural field

- Model the signed distance field $u(x, y, z) = MLP_\theta(x, y, z)$ with $\theta$ 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]

# Implicit neural field

- Model the signed distance field $u(x, y, z) = MLP_\theta(x, y, z)$ with $\theta$ 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]

# Implicit neural field

- Model the signed distance field $u(x, y, z) = MLP_{\theta}(x, y, z)$ with $\theta$ 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 $(x_i, \mathsf{n}_i), i \in I$
- Look for $u$ continuous and a.e. $\mathcal{C}^1$ such that:

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

- Loss [Gropp 2020]

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

# Periodic Activation Functions [Sitzmann 2021]

- Replace ReLU by periodic activation function $\rightarrow$ 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 $\psi(u_\theta(x)) = \exp -\alpha|u_\theta(x)|;\ \alpha >> 1$

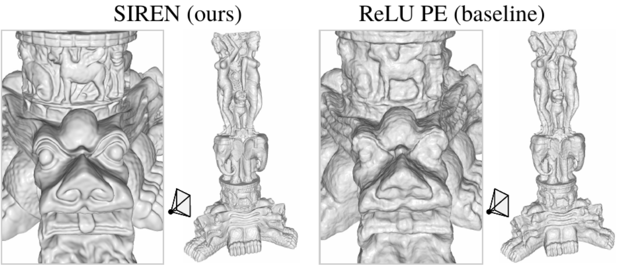



Figure 4: A comparison of SIREN used to fit a SDF from an oriented point clouse against the same fitting performed by an MLP using a ReLU PE (proposed in [35]).

From [Sitzmann 2020]

# Periodic Activation Functions [Sitzmann 2021]

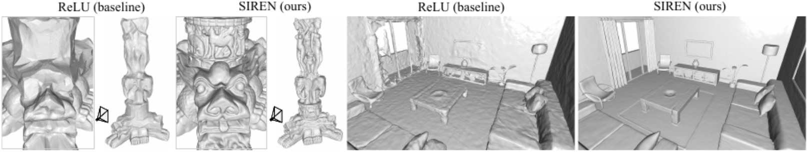
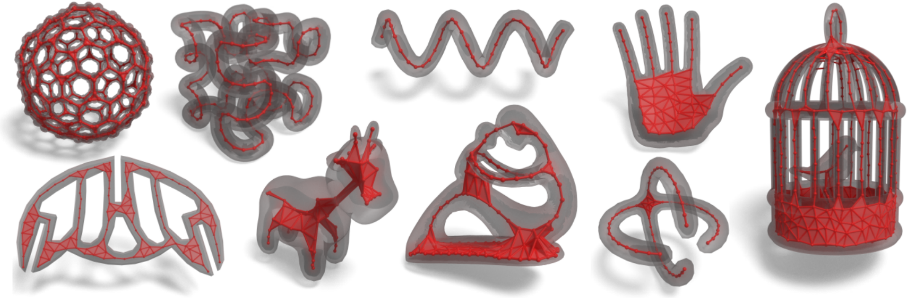



Figure 4: Shape representation. We fit signed distance functions parameterized by implicit neural representations directly on point clouds. Compared to ReLU implicit representations, our periodic activations significantly improve detail of objects (left) and complexity of entire scenes (right).

From [Sitzmann 2020]

# Regularizing INR away from the surface

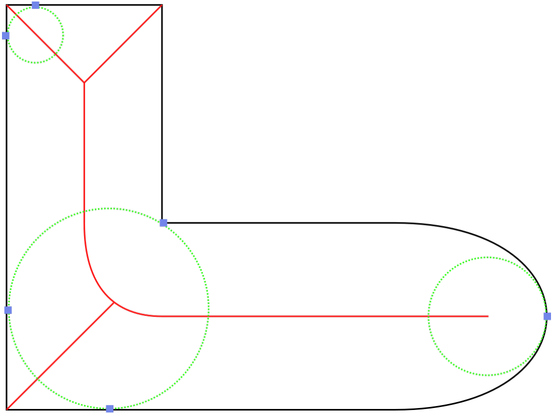


[Clémot, Digne 2023]

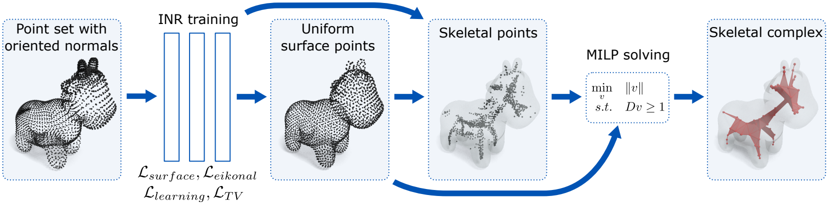
# Medial Axis

## 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

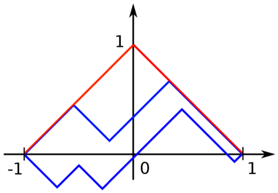

Point set with oriented normals
INR training
$\mathcal{L}_{surface}, \mathcal{L}_{eikonal}$
$\mathcal{L}_{learning}, \mathcal{L}_{TV}$
Uniform surface points
Skeletal points
MILP solving
$\min_v \|v\|$
$s.t.\ Dv \geq 1$
Skeletal complex

# 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



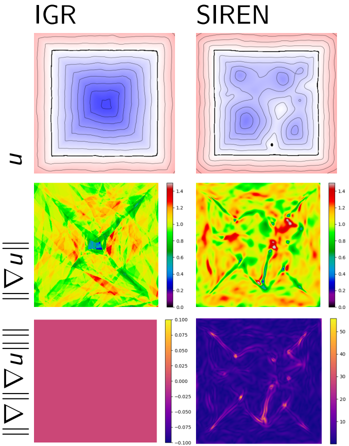

[Camilli 2014]

### 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

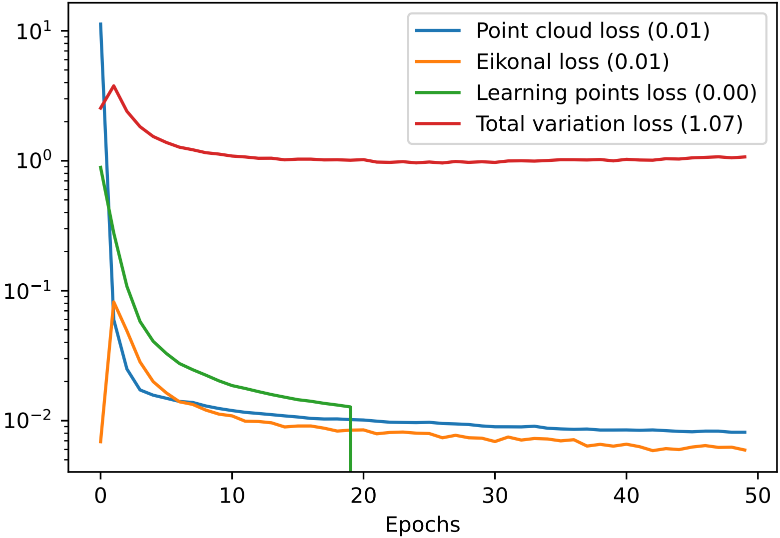
$$\mathcal{L}_{\text{learning}} = \sum_{p \in \mathcal{P}} (u(p) - d(p))^2 + \sum_{p \in \mathcal{P}} 1 - \frac{\nabla u(p) \cdot \nabla d(p)}{\|\nabla u(p)\| \, \|\nabla d(p)\|} \tag{4}$$

- + TV loss

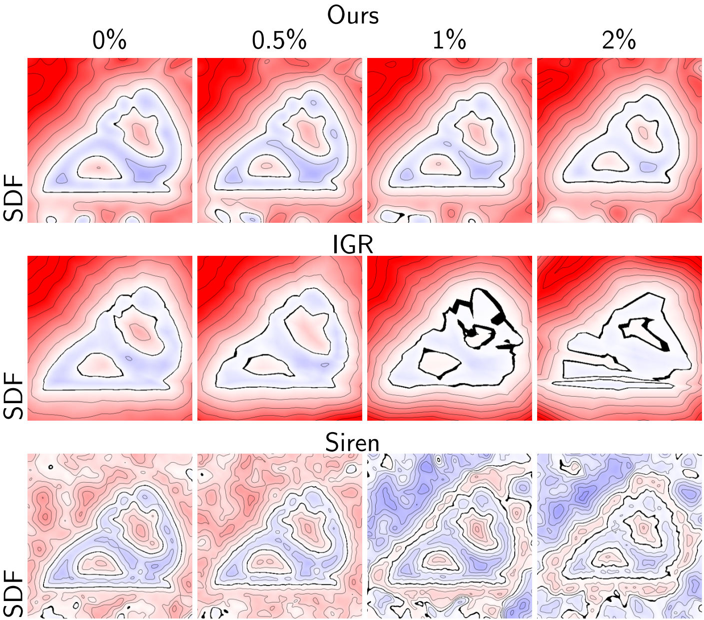
## Loss

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

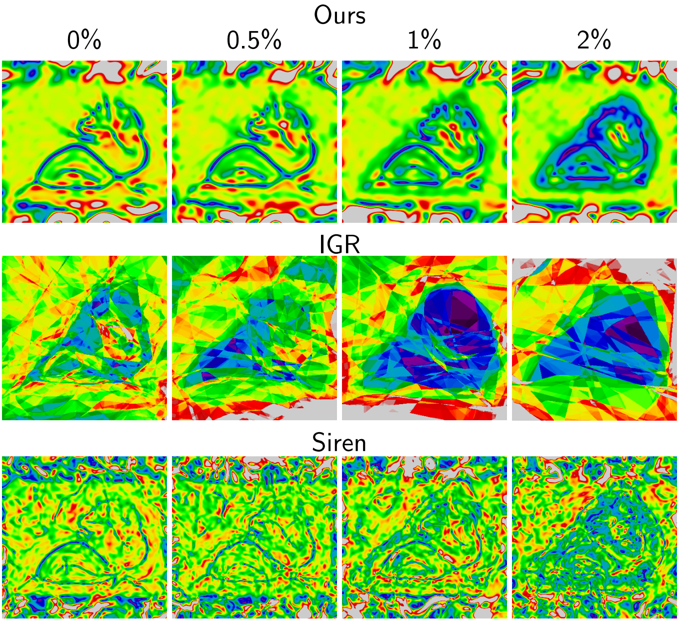
# Convergence



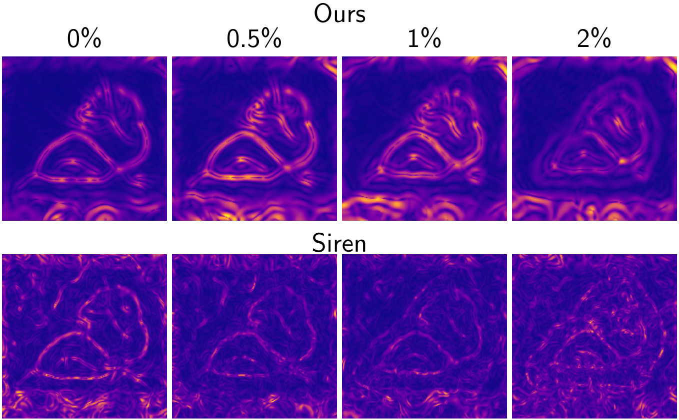
# Resulting Fields



# $\|\nabla u\|$



# $\nabla \|\nabla u\|$



# 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: $D$ $(N \times M)$

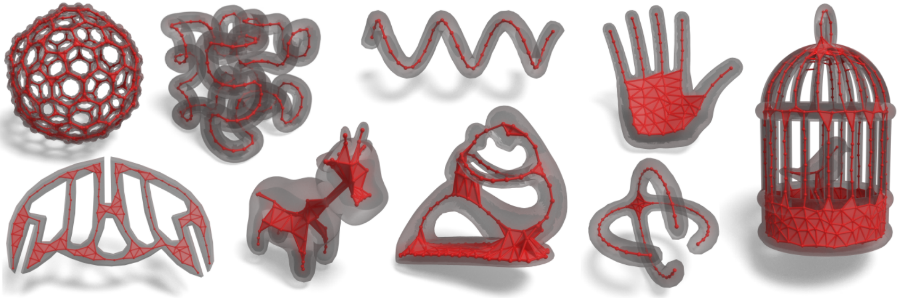
$$D_{ij} = 1 \text{ if } \|p_i - s_j\| - r_j \leq \delta \text{ and } 0 \text{ otherwise}$$

  - Mixed Integer Linear Problem:

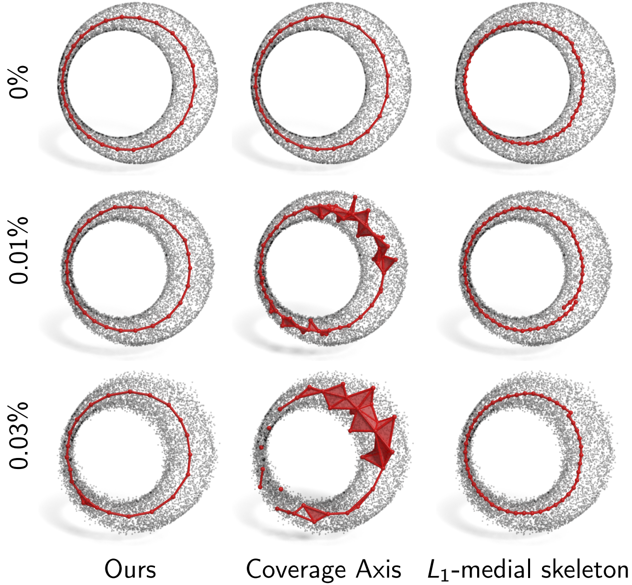
$$\begin{array}{ll} \min & \|v\|_2 \\ \text{s.t.} & Dv \succeq 1 \end{array} \tag{6}$$

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

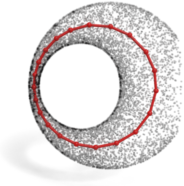
# Results



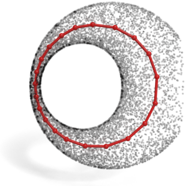
# Results



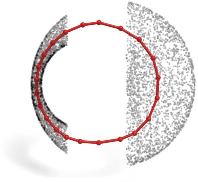
# Results



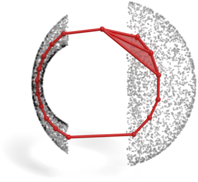
Ours



Coverage Axis

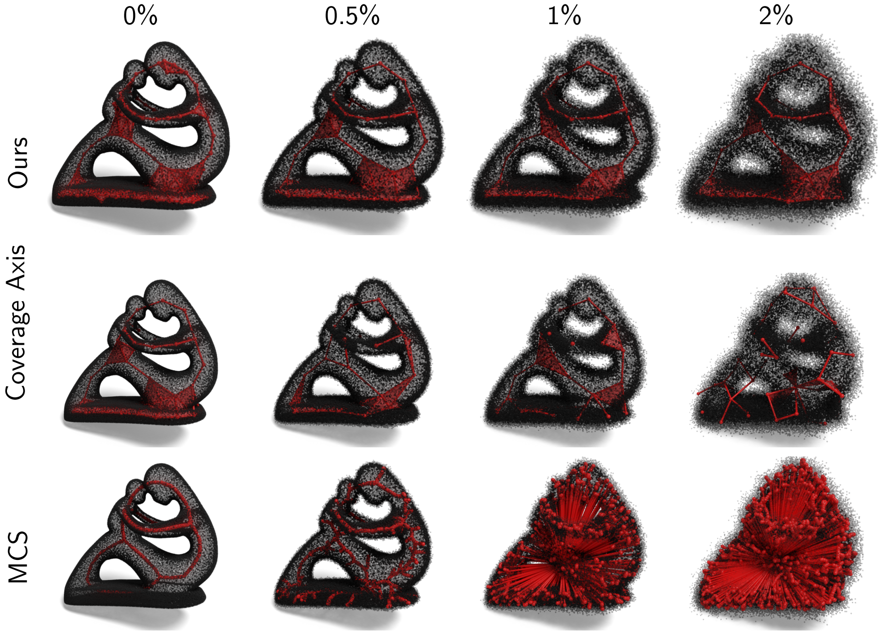


Ours

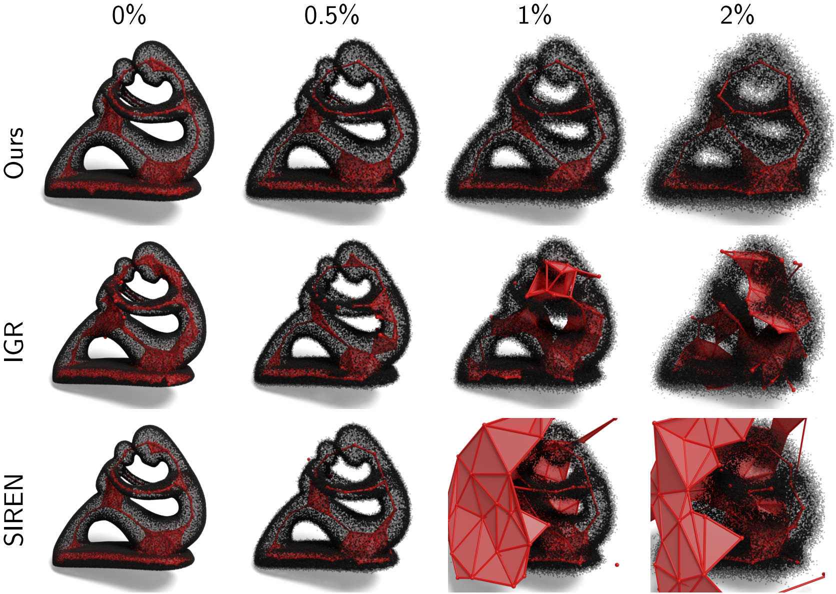


Coverage Axis

# With noise



# With noise



# 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).

# Conclusion

- Shape reconstruction is a long standing problem: no universal solution.
- Benefits from advances in Machine Learning and Optimization.
- Do we need to reconstruct the surface? Or just be able to render it?