Computers & Graphics 114 (2023) 368–378

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on SMI 2023

# Neural skeleton: Implicit neural representation away from the surface

Mattéo Clémot *, Julie Digne

*LIRIS, CNRS, Université Lyon 1, France*

ARTICLE INFO

*Article history:*
Received 17 May 2023
Received in revised form 7 June 2023
Accepted 14 June 2023
Available online 17 June 2023

*Keywords:*
Skeleton computation
Neural implicit representation
Total variation

ABSTRACT

Implicit Neural Representations are powerful tools for representing 3D shapes. They encode an implicit field in the parameters of a Neural Network, leveraging the power of auto-differentiation for optimizing the implicit function and avoiding the need for a manually crafted function. So far, Implicit Neural Representations have been mainly designed to extract or render object surfaces and methods primarily focus on improving the implicit function near the surface. In this paper we argue that implicit fields are useful for other shape analysis tasks, in particular skeleton (medial axis) extraction. Indeed, a medial axis is defined through distances to the surface, which can be provided by an implicit neural representation, making it robust to noise and missing data. However this requires the implicit field to be reliable away from the surface, something most representations are not optimized for. To achieve this, inspired by variational image denoising techniques, we propose to add a Total Variation term, to regularize the implicit field. We further design a skeleton sampling method working directly on the GPU, and link the extracted points using a coverage formulation. We show that our resulting neural skeleton is more robust to sample defects such as noise or missing data compared to other medial axis extraction methods.



## 1. Introduction

Neural fields are a growing trend of today's computer graphics and computer vision research. They are used for a wide variety of tasks, for scene reconstruction and novel viewpoint rendering (see [1] for a comprehensive survey).

Implicit surface representation has been a part of the graphics pipeline for decades, be it for geometric modeling [2] or for surface reconstruction [3] with efficient methods for extracting a mesh [4] or direct rendering using e.g. sphere tracing [5]. Recently it has gained a lot of traction, by using a neural network for the implicit function instead of a manually crafted one. This *de facto* encodes the shape into the parameters of a neural network, to be optimized over the input points, either from a single shape, or over a latent shape space [6,7]. This formalism has proven to be very efficient for extracting object or scene surfaces from an input point set, possibly overcoming noise, outliers and missing data. So far, focus has been directed towards improving the neural field near the surface, the ultimate goal being to improve the surface rendering or mesh extraction.

In this paper, we instead propose to improve the neural field away from the surface. By doing so, we aim at extracting topological information that is contained in the signed distance field of the shape. The medial axis is one of such topological characteristics. It is formally defined as the locus of points that have at least two nearest points on the surface. From a signed distance field perspective, it corresponds to distance gradient discontinuities. However this characterization requires a reliable signed distance estimation, especially near the medial axis, where many implicit neural representations (INR) exhibit artefacts: the gradient norm changes, while it should remain constant. To alleviate this, we propose to constrain the variations of the norm of the gradient. We draw our inspiration from image regularization techniques where the Total Variation (TV) allows to get smooth color gradients [8]. By penalizing high total variation of the gradient's norm, we constrain the norm to be as-constant-as-possible which yields a much more stable signed distance field.

In summary, our contributions are as follows:

- A TV regularization term yielding a smoother signed distance function away from the surface.
- A skeleton sampling method on the GPU.
- A complete pipeline to extract a medial complex from an input point cloud.

## 2. Related work

### 2.1. Implicit neural representations

Analyzing 3D data using machine learning techniques has been a challenge tackled by many researchers in the past decade.

* Corresponding author.
*E-mail addresses:* matteo.clemot@ens-lyon.fr (M. Clémot), julie.digne@cnrs.fr (J. Digne).

https://doi.org/10.1016/j.cag.2023.06.012
0097-8493/

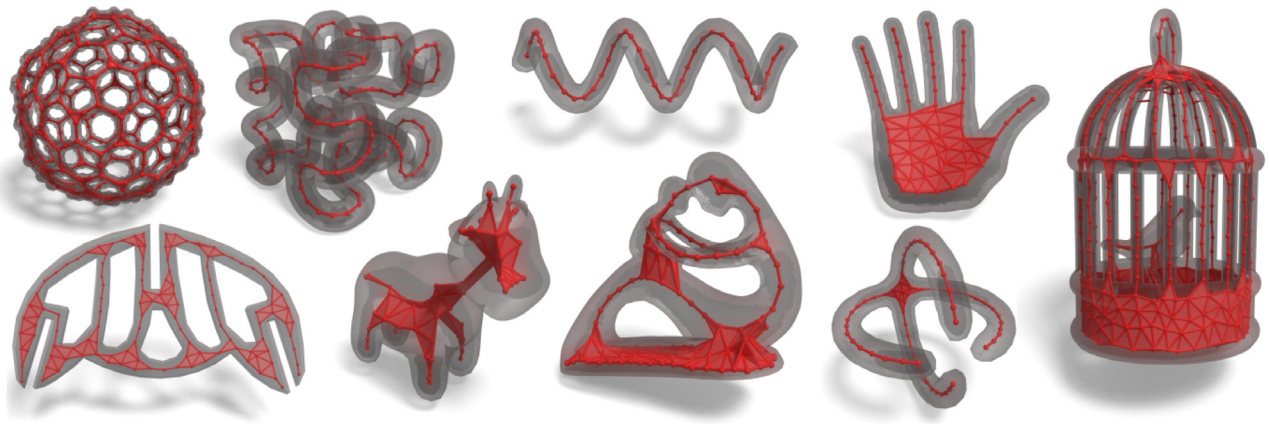

**Fig. 1.** Neural Skeletons computed on several shapes with various topologies.

Center to this challenge is the question of shape representation: how to represent geometric data, for it to be amenable to deep learning [9,10]. Recently, implicit representations have gained a lot of traction, not only for surface reconstruction but also for novel view rendering. We focus here on the main methods for shape representation and refer the reader to [1] for a broader state of the art of this very active research field.

Pioneering this representation, Park et al. [6] introduced an auto-decoder able to optimize the latent code of a shape using MAP estimation and estimate the Signed Distance Function (SDF) at any point in the ambient space, given the shape latent code and the query point coordinates. Indicator functions have also been used instead of SDF for shape reconstruction [7] or for shape generation [11]. Indicator functions are interesting for they allow to focus on the surface itself without learning accurate distance values anywhere in the ambient space.

Following the neural novel view rendering trend [12] which encodes a scene radiance field in the parameters of a multilayer perceptron optimized in a self-supervised way, INRs have been used for encoding a signed or unsigned distance field or an occupancy indicator, always with a view of reconstructing the surface. These networks are trained per shape and do not rely on a database. Atzmon and Lipman [13,14] and Chibane et al. [15] learn an SDF from raw data, point clouds or triangle soups, without needing an oriented normal, hence alleviating the need for a groundtruth sign or watertight surfaces. Gropp et al. [16] proposed a new way of optimizing the SDF, with an eikonal loss enforcing a unit norm for the SDF gradient and a surface loss forcing the SDF to vanish at the input surface samples. Many further developments were made: Peng et al. [17] mixed a voxel-based convolutional encoder with an implicit decoder to be able to handle large scale data. Alternatively, one can encode a large scale shape through only local implicit fields instead of a global one to handle details and remain light-weight [18,19]. By changing the activation from ReLU to periodic functions, Sitzmann et al. [20] introduced SIREN which allowed to better recover shape details and compute reliable gradients. Interestingly, SIREN is a very general formulation, also useful for solving a wide variety of partial differential equations. It was later used in a multiscale way to decompose the shape into a smooth shape and an implicit displacement field encoding the details [21], yielding unparalleled detail recovery. Many improvements have later focused on levels of details [22], latent space regularization [23]. For surface reconstruction or rendering, it is also possible to learn only a potential, Lipman [24] proposed to borrow from fluids phase transition to learn an implicit function converging to an occupancy function, and Williams et al. proposed a kernel-based INR [25] to reconstruct a potential. However all of these methods target shape surface extraction and focus their quality effort around this level set.

INR being optimized per shape, speed can be an issue. To alleviate this, Müller et al. introduced InstantNGP [26] which is able to compute an SDF in a few seconds. While this is crucial for fast visualization, our analysis task does not require to be real time, and we do not follow this path.

Dedicated methods for rendering implicit representations, without going through a mesh extraction, have been proposed, way before the rise of deep learning. The most standard rendering method remains sphere tracing [5] which casts rays along a viewing direction and samples point on it depending on the signed distance field local value.

Neural implicits have also been recently employed for other tasks than visualization or shape reconstruction, for example to perform guaranteed queries on shapes [27], but not for medial axis extraction.

### 2.2. Skeleton extraction

Skeleton extraction from geometric data is a well researched problem, we refer the reader to Tagliasacchi et al.'s survey [28] for an extensive description of the different approaches and focus here on the main ones. In a nutshell, skeleton extraction consists in extracting a graph-like structure able to summarize the shape. Several mathematical definitions for the skeleton exist, including curve skeleton and medial axis. A curve skeleton represents the shape through a set of curves. It has been formalized using medial geodesic functions [29], although most methods do not use this definition and remain empirical. It has gathered a lot of interest in particular due to its proximity to animation skeleton. Curve skeletonization techniques include mesh contraction [30], using the visual hull [31], or using a Mean Curvature Flow [32]. It can also work directly from raw point clouds [33], leveraging local symmetries and normal information to compensate for missing data. Using the $L_1$ norm also allows to be more robust to outliers in point cloud data [34].

Shapes often contain non-tubular parts, which curve skeleton-based methods have trouble handling. In the latter case, it may be more interesting to use the medial axis. The medial axis is mathematically defined as the set of points having more than one closest point on the surface [35]. It has been extensively studied in computational geometry [36–38]. Finding compact medial axis representation has been tackled from an error minimization perspective [39,40], by minimizing the reconstruction error from the medial axis and associated maximally inscribed sphere radii. Surrogates for medial axis transform have also been studied such as Deep Point Consolidation which optimize a set

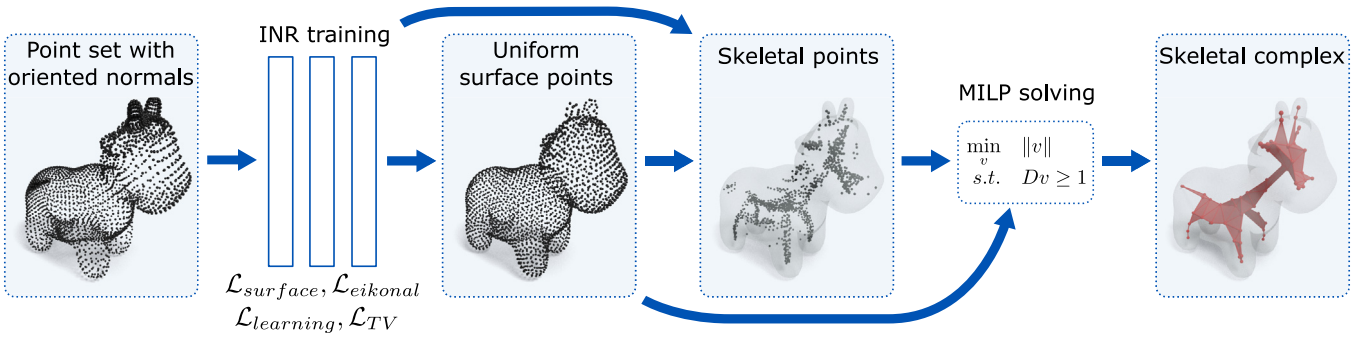


**Fig. 2.** Overview of our method. An implicit neural representation is trained on the input point set, uniform surface points and skeletal points are extracted, and the skeleton mesh is recovered using a cover-set formulation solved as a Mixed-Integer Linear Program (MILP).

of inner points on a meso-skeleton [41], however this method requires a heavy machinery. Yan et al. [42] describes a medial axis extraction method based on shape voxelization and Voronoi diagram extraction of the boundary voxels. This method has the advantage to come with theoretical guarantees. Our method avoids the voxelization step and relies instead on the regularized signed distance field to extract the medial axis.

Recently, Dou et al. [43] proposed a method to extract the medial axis of meshes or point clouds by formulating the problem as a cover-set problem. This allows to recover compact and meaningful medial axes, very efficiently. This method relies on a set of good candidate skeletal points which our method can provide in a more robust way, as will be shown in the experiments. We will compare our results to those classical approaches.

### 2.3. Deep learning and skeleton extraction

Animation skeleton computation has been tackled from a deep learning perspective [44], but few methods have tackled medial axis learning from a dataset. Point2Skeleton [45] uses a PointNet encoder [9] to synthesize skeletal point, and predict their links using a graph auto-encoder. However the skeletal points are not mathematically defined and the losses only encourage a medial-axis like position. Furthermore it requires to train on a shape database and is therefore limited to shapes that fit in the learned latent space. P2MAT-NET [46] estimates the set of medial points and medial spheres using a ground truth skeleton-shape dataset and links the medial points using a Delaunay triangulation which is later pruned. Both these methods are very different from our approach which works on a single shape and does not require a training dataset. In a quite different direction, it has been argued that taking into account the medial axis for shape recognition leads to better recognition results [47]. In practice it involves computing a set of medial spheres, using a computational geometry technique, and feeding it to a neural network.

Skeleton extraction from a neural implicit perspective, without relying on a database, has not yet been tackled yet. Deep Medial Fields [48] was the first paper to link neural fields and medial axis transform, by estimating a medial field. For each point in the ambient space, it evaluates the shape width (i.e. the distance between the surface and the medial axis) in the "slice" containing the point. However this medial field is discontinuous near the surface, and two neural networks are used for handling this discontinuity. While it allows to design an adapted sphere tracing algorithm, it is not applied to skeleton extraction.

## 3. Overview

Input to our method is a set of points, sampled on the surface of an object, which we assume to be closed. The points are supposed to be endowed with an oriented normal. If none is provided, we compute the normals by local PCA and orient them using a minimum spanning tree [3]. Our goal is to estimate the medial axis of the object, defined as the set of inner points which lie at equal distance from at least two points of the surface. Our key insight is that leveraging the power of INR, and regularizing them adequately, can help dealing with missing data and noise. Our method is summarized on Fig. 2. We proceed in three steps:

- First, we estimate a signed distance field by adding an extra total variation regularization term to a sine-activation network.
- Then, we extract skeletal points by a dedicated skeleton sampling method on the GPU.
- Finally, we construct the skeletal simplicial complex by using a mixed integer programming solver, following the Coverage Axis method [43].

## 4. Mathematical background

### 4.1. The eikonal PDE

Signed distance functions and solutions of the eikonal equations are an intensively studied problem in image analysis, in particular for their links with fast marching [49]. The signed distance function is a solution to the eikonal equation with mixed Dirichlet and Neumann conditions. For the sake of this analysis, let us first assume we are dealing with an object $\Omega$ with a $\mathcal{C}^1$ boundary (with normal $\mathbf{n}_\Omega$) and positive reach.

The problem is to look for a continuous and almost everywhere differentiable function $u$ such that:

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

From a partial differential equation point of view, this equation has an infinite number of weak solutions by adding gradient singularities. The viscosity theory allows to discriminate between those solutions and shows that the unique viscosity solution to the eikonal equation is the signed distance function [50]. Hence, finding $u$ that minimizes only an eikonal loss is not enough to recover a signed distance field [16].

Theoretically, the solution $u$ to Eq. (1) should be in Sobolev space $W^{1,p}$ [50]. Interestingly Lipman [24] showed that, under appropriate Lipschitz boundary constraints for the subset of $\mathbb{R}^3$, the function resulting from a trained MultiLayer Perceptron (MLP) with Rectified Linear Units (ReLU) activation functions is in this exact Sobolev space. However INRs with ReLU activation functions are sometimes hard to optimize. Furthermore, differentiation with respect to the input coordinates is unstable due to the non-differentiability of ReLU. To alleviate this, Sitzmann et al. [20]

introduced a periodic activation function replacing ReLU in the MLP leading to infinitely differentiable functions. While this is a desirable property for solving regular PDEs, it theoretically cannot represent a function which has gradient singularities, such as a SDF. Fortunately, the singularities happen away from the surface $\partial\Omega$, and have low impact on the surface extraction itself. To improve the estimation far from the surface, we propose to add a regularization term to the loss.

### 4.2. Total variation of the gradient norm

SDFs are not only useful for surface reconstruction they also encompass a lot of topological information about the shape. In particular the medial axis corresponds exactly to the gradient singularities inside the shape. Since a sine-network gradient cannot be discontinuous, by construction, discontinuities will be replaced by 0 values for the gradients. Our goal is to make this 0 gradient as localized as possible, to get an accurate medial axis. Our insight is that if, in addition to being close to 1, we encourage the norm of the gradient to remain as constant as possible, we will get a better axis localization. Therefore, we want to render the norm of the gradient as constant as possible.

Inspired by variational image regularization approaches, we regularize the field by using the Total Variation of the norm of the gradient. The Total Variation of a scalar field measures how much this scalar field varies over the domain: a near constant scalar field will have a Total Variation close to 0 [8]. We apply this principle to the norm of the gradient which should remain constant almost everywhere and formalize it as:

$$\int_{\mathbb{R}^3} \|\nabla\|\nabla u\|\| = 0 \tag{2}$$

Notice first that the true SDF naturally satisfies this condition since the norm of its gradient is 1 almost everywhere (everywhere the gradient is defined), the $\mathbb{R}^3$-Lebesgue measure of the skeleton being 0. Hence this regularization does not push us away from the true solution, it just adds a term that is trivially satisfied by the true SDF.

**Theorem 4.1.** *The TV term favors that u has no order 2 differential content along its gradient lines.*

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

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

with $H_u$ the Hessian matrix of $u$. $H_u \frac{\nabla u}{\|\nabla u\|}$ corresponds to the order 2 differential content of $u$ in the gradient direction. In other words gradient should remain constant along gradient lines. □

An intuitive way of looking at the TV regularization term is by remembering that it provides a quantity which is roughly proportional to the measure of the gradient's discontinuities, that we are trying to minimize. Although it is possible to build solutions of the eikonal equation that have smaller set of discontinuities in the Lebesgue sense than the medial axis, the hope is that these solutions will be avoided by adding enough learning points (points whose distance and gradient are roughly known). In practice we see that the TV loss improves the distance and gradient fields (see Section 10).

## 5. Total variation regularized implicit neural representation

We implement the TV constraint by adding a term to the loss of a Sine-activation based network [20], which, once optimized, takes an input 3D position $p$ and outputs the SDF values at $p$. One of the advantage of using Sine-activation is that the network produces infinitely differentiable functions. This makes the computation of $\nabla u$ and $\nabla\|\nabla u\|$ trivial using auto-differentiation. On a side note, periodic functions also improve the training convergence of the network. However, the downside is that we will not estimate the true SDF but an infinitely differentiable surrogate for it. The optimization should make it as close as possible to the true SDF.

In practice we use a network with 6 layers of size 64, and train it using the following losses:

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

$$\mathcal{L}_{\text{surface}} = \int_{\partial\Omega} u(p)^2 dp + \int_{\partial\Omega} 1 - \frac{\mathbf{n}(p)\cdot\nabla u(p)}{\|\mathbf{n}(p)\|\,\|\nabla u(p)\|} dp \tag{4}$$

$$\mathcal{L}_{\text{learning}} = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}(u(p) - d(p))^2 + \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} 1 - \frac{\nabla u(p)\cdot\nabla d(p)}{\|\nabla u(p)\|\,\|\nabla d(p)\|} \tag{5}$$

The eikonal loss (Eq. (3)) favors a solution to the eikonal PDE. The surface loss (Eq. (4)) ensures that the estimated signed distance function is 0 at the surface points, and that its gradient aligns well to the surface normals. The learning points loss (Eq. (5)) takes into account a set $\mathcal{P}$ of points sampled in the ambient space for which we know the true signed distance $d(p)$ and its gradient $\nabla d(p)$. These points ($1000 - 10000$ depending on the complexity of the shape) are sampled uniformly in the bounding box. In practice, this ground truth value is approximated by the signed distance to the nearest point of the input point cloud. This value can easily be wrong for partial point sets, and, to alleviate this, we use this learning points loss only in the first steps of our training iterations.

To these losses, we add our total variation regularization term, explained above:

$$\mathcal{L}_{\text{TV}} = \int_{\mathbb{R}^3} \|\nabla\|\nabla u\|(p)\| dp \tag{6}$$

The total loss writes:

$$\mathcal{L} = \lambda_e\mathcal{L}_{\text{eikonal}} + \lambda_s\mathcal{L}_{\text{surface}} + \lambda_l\mathcal{L}_{\text{learning}} + \lambda_{\text{TV}}\mathcal{L}_{\text{TV}} \tag{7}$$

The loss weighting parameters $\lambda_e, \lambda_s, \lambda_l, \lambda_{TV}$ are set manually to keep the ranges comparable. In our experiments we used the following values:

$$\lambda_e = 100 \quad \lambda_s = 100 \quad \lambda_l = 100 \quad \lambda_{\text{TV}} = 20.$$

While different weight scheduling (e.g. decreasing one of the parameters over time) could be devised, we found experimentally that keeping these weights constant provided good results, except the learning points loss weight which is set to 0 after 20 epochs.

We use LBFGS [51] as an optimizer, which we found more efficient than ADAM. We stop the training when the loss variation becomes negligible. It takes 50 iterations and 60 s with LBFGS, against 20000 iterations and 500 s with Adam. While LBFGS is more complex and each iteration costs more, it needs fewer iterations and a lower overall computation time. Our network is pre-trained on the SDF of a sphere with radius 0.5, as advised by Gropp et al. [16], and all shapes are normalized in a unit-length cube centered at 0. This pre-training speeds up the convergence of the network.

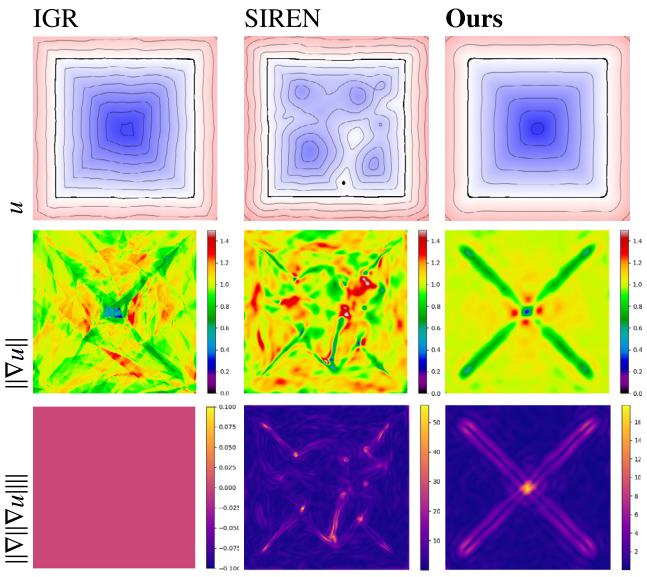


**Fig. 3.** Slices of the SDF, gradient's norm and norm of the gradient's norm, using IGR, SIREN and our TV-regularized method on a cube. IGR using ReLU activation function, the second order derivative is 0 almost everywhere.

Fig. 3 shows some slices of the obtained signed distance field with and without TV regularization. The TV regularized SDF yields a norm of the gradient with less variations leading to a better SDF.

Once our implicit neural representation is trained, we compute the skeleton mesh in three steps: we first sample points uniformly on the surface, we then recover skeletal points, and finally solve a cover set problem [43] to build the final skeleton simplicial complex.

## 6. Surface points

The next step consists in sampling the neural surface uniformly, which will enable us to both obtain skeletal points (Section 7) and represent well the surface when selecting those skeletal points using a coverage formulation (Section 8). We follow the Iso-Points method [52]. First, we sample points uniformly in the object's bounding box and move them towards the 0 level set using Newton's method. Starting with $N$ random points in the ambient space, we iterate, for each point:

$$p \leftarrow p - \frac{\nabla u(p)}{\|\nabla u(p)\|^2} u(p) \tag{8}$$

Then $p$ converges to a point on the 0 level set of the implicit function, which – provided the implicit function was well constructed – should be the object's surface.

However, at the end of this process, the distribution of these projected points $(p_i)_{i=1\cdots N}$ is in general not uniform. To uniformize it, we iteratively take these points away from dense regions:

$$p \leftarrow p - \alpha r \tag{9}$$

where the step size $\alpha$ is set to $\sqrt{D/N}$ with $D$ the length of the diagonal of the bounding box of the shape. The step direction $r_i$ for a point $p_i$ is given by the weighted barycenter of the directions to its $K$ nearest neighbors projected on the tangent plane $\mathcal{T}$:

$$r_i = \sum_{p_j \in \mathcal{N}_K(p_i)} w(p_i, p_j) \Pi_{\mathcal{T}} \left( \frac{p_j - p_i}{\|p_j - p_i\|} \right). \tag{10}$$

where $\Pi_{\mathcal{T}}$ is the orthogonal projection operator on the tangent plane $\mathcal{T}$. The weight function permits to decrease the influence of further neighbors and is set to

$$w(p_i, p_j) = \exp\left( -\frac{\|p_i - p_j\|^2}{\sigma} \right) \text{ with } \sigma = 16D/N. \tag{11}$$

After this resampling, the points may have drifted away from the surface, but this drift is limited by the restriction to the tangent plane. Notice that the original Iso-Point method does not restrict this motion to the tangent plane. To reduce this drift further, we project the points back onto the surface using again Newton iterations. By iterating projection and uniformization, we converge to a uniform sampling of the surface.

## 7. Skeletal points

To extract the skeletal points, we make the simple observation that if we cast a ray from a surface point in the opposite direction to the gradient, then this ray crosses the medial axis before hitting the surface again. In addition, at the medial axis crossing, the SDF should be minimum (assuming the SDF to be positive outside and negative inside the object). This translates into a simple skeleton tracing algorithm.

For each surface point $p$, obtained as described in Section 6, we compute its gradient $\nabla u(p)$ by auto-differentiation, and sample $n$ points $q_i$ regularly on a line starting from $p$ in the direction of $-\nabla u(p)$ until some fixed distance $h$. We extract the first point $q_{i_0}$ with a positive SDF value along this line, meaning we have crossed the surface and are again outside of the object. Then we resample segment $pq_{i_0}$ regularly and take the point where the gradient has the smallest norm. This sampling algorithm involves only linear operations and SDF queries, hence it can be done directly on the GPU, by dividing the initial random points into batches to avoid memory issues. The method is summed up in Algorithm 1.

In practice we take $h = 0.5$, since all shapes are normalized within a cube with side 1. We also set $N = 10000$ initial points and $n = 50$ ray subdivision.

**Data:** An estimated SDF $u$, $N$ and $n$ two integer parameters, $h$ maximum search distance.
**Result:** A set of skeletal points
Sample $N$ points $(p_i)$ on the surface using Newton's method;
**for** $i = 1 \cdots N$ **do**
  Sample $n$ points $q_j = p_i - t\frac{h}{n}\frac{\nabla u(p_i)}{\|\nabla u(p_i)\|}$ $(t = 1 \cdots n)$;
  Find $i_0$ the smallest index such that $u(q_{i_0}) > 0$;
  Sample $n$ points $r_j$ between $p$ and $q_{i_0}$;
  Find $r_j$ with lowest $\|\nabla u(r_j)\|$ value;
**end**

**Algorithm 1:** Skeleton sampling

The skeletal points obtained with Algorithm 1 can be non-uniform and uselessly dense in some areas. Therefore, in order to speed up the resolution of the MILP in the next step, we eventually subsample the skeletal points using a simple random procedure: we iteratively randomly select a skeletal point and remove all skeletal points within a given distance to it.

## 8. Skeleton mesh extraction

To extract the skeleton mesh from our skeleton points, we follow the Coverage Axis method [43]. This method selects a subset of our skeletal points by formulating the coverage problem as a mixed integer linear problem (MILP), and links them using a weighted triangulation.

The coverage axis point selection method starts by considering a set $S$ of $M$ skeletal candidate points with their medial sphere radius and a set $P$ of $N$ surface points. In our case we take

our (subsampled) skeletal points $s_i$ along with their estimated distance $r_i = u(s_i)$ and the uniformly sampled surface points $p_i$. We then build a MILP, whose goal is to select a subset of the skeletal points with the constraint that their skeletal ball should cover all the surface points.

Let $v$ be a vector of size $M$ such $v_i = 1$ if the skeletal point $s_i$ is selected and 0 otherwise. We construct $D$ a $N \times M$ matrix such that $D_{ij} = 1$ if $\|p_i - s_j\| - r_j \leq \delta$ and 0 otherwise. The selection procedure writes as the following MILP:

$$\begin{aligned} \min \quad & \|v\|_2 \\ \text{s.t.} \quad & Dv \geq 1 \end{aligned} \tag{12}$$

In practice we rely on a MILP solver with a time limit set to 1000 s. While it does not always allow to reach optimality for complex models, we observe that the lower bound is very close (a few vertices) to the solution, making it near-optimal. Importantly enough, each solve might give a slightly different solution.

Eventually, the selected skeletal points are meshed by computing the regular triangulation of the union of the selected skeletal points (with weight $r_j$) and the surface points (with weight $\delta$), i.e. $\{(s_j, r_j), s_j \in S \mid v_j = 1\} \cup \{(p_i, \delta), p_i \in P\}$. We then extract the edges and triangles between the selected skeletal points that appear in this regular triangulation.

## 9. Experiments and comparisons

Our code is available at https://github.com/MClemot/SkeletonLearning. We run all our experiments with a Nvidia RTX 3050 laptop GPU. We use the PyTorch library for our network implementation, the Gudhi library [53] for the regular triangulation and SciPy for the MILP solver [54].

We test our algorithm on shapes from Thingi10K [55] and AIM@shape. We are interested in particular in shapes that exhibit a high genus and a lot of topological details such as the ones of Figs. 1 and 5. When the input is a mesh, we sample $100k$ points uniformly on the mesh triangles, following the method of Osada et al. [56]. We compare our skeleton with $L_1$-medial skeleton [34], and VoxelCores [42] using the authors' code, Mean Curvature Skeleton [32] using the CGAL implementation, and reimplemented Coverage Axis [43].

Fig. 4 compares our neural skeletons with several other methods on standard shapes. On noiseless shapes, all methods give good results. $L_1$-medial skeleton works well on tubular shapes but is less efficient on shapes that are not tubular. VoxelCores provides a skeleton which has the right topology without any regard for its compactness. As such it contains much more mesh simplices. Fig. 5 compares our Neural Skeleton with VoxelCores and Coverage Axis on high-genus, non-tubular shapes. VoxelCores produces a non compact but correct skeleton, Coverage Axis produces skeletons whose vertices are sometimes outside of the shape boundary and the MILP solver connects points that should not be connected (on the birdcage). In contrast, our skeletal point extraction is much more accurate and leads to less artifacts in the skeletons.

Fig. 6 further shows the wrong connection issues that can appear with the Coverage Axis initial candidate skeletal points. The MILP will only select a subset from these skeletal points but will not correct their positions, hence the importance of having a good set of candidate skeletal points, as those provided by our approach.

The VoxelCores pipeline uses a signed distance field built using PolyMender [57], which does not handle point clouds, MCS also relies on a reconstruction step. On the contrary, Coverage Axis, $L_1$-medial skeleton and our method can work with partial point sets, as shown on Fig. 7. Our method is less sensitive to missing data, and matches the shape feature better.

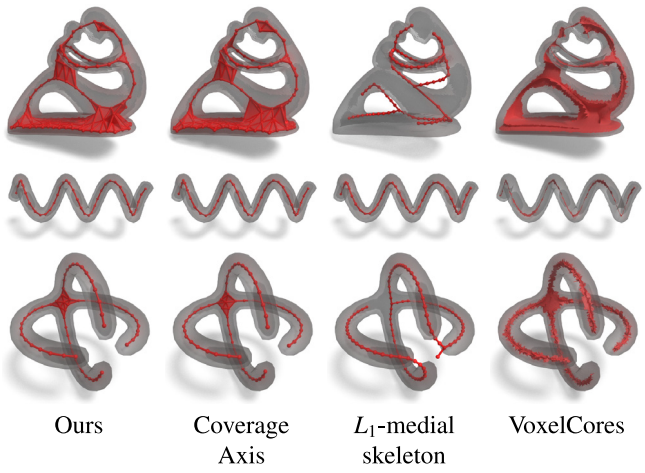


**Fig. 4.** Comparison with Coverage Axis, $L_1$-medial skeleton and VoxelCores on standard shapes.

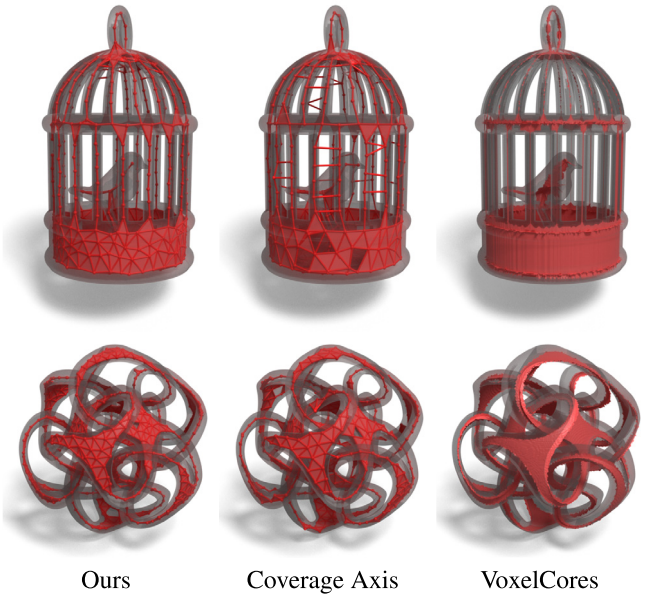


**Fig. 5.** Comparison with Coverage Axis and VoxelCores. Our method produces more compact skeletons than VoxelCores with less artifacts (wrong connection edges) as Coverage Axis.

Fig. 8 shows a point cloud with increasing noise and compare our result with Coverage Axis and $L_1$-medial skeleton. While for low levels of noise all methods perform well, as noise increases the performance of Coverage Axis degrades. Fig. 9 shows how our method behaves with missing data. To a certain extent our method is able to fix missing parts, which is due to the constraints in the neural implicit optimization.

Apart from these traditional methods, we compare to baselines using other INR. We replace our TV-regularized INR in our pipeline, by standard INR such as Implicit Geometric Regularization (IGR) [16] or SIREN [20]. Fig. 12 shows the different results on the fertility shape with increasing levels of noise, while Fig. 13 shows 2D slices of the SDF and its derivatives, compared to other INRs, on the same fertility shape with increasing levels of noise. As expected, the slices of the derivatives are much smoother and stabler with our method. We also compare numerically all these methods on a synthetic shape with noise and missing data in Table 1. The distances are computed by sampling the ground truth

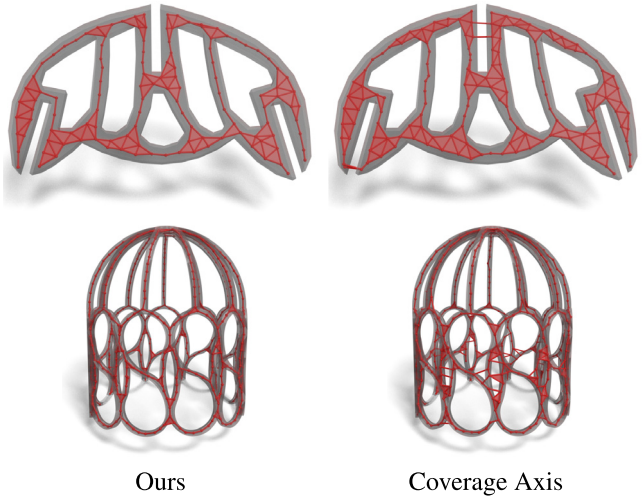


**Fig. 6.** Comparisons with Coverage Axis on some complex shapes.

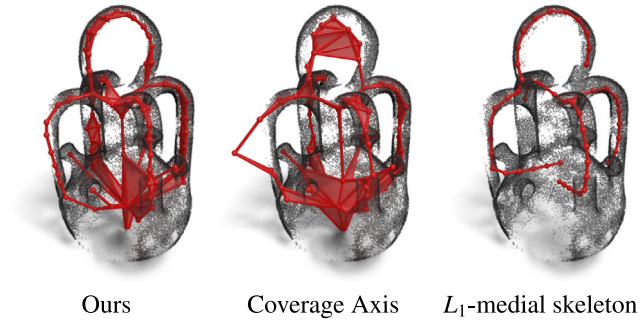


**Fig. 7.** Comparison with Coverage Axis and $L_1$-medial skeleton on a partial point set.

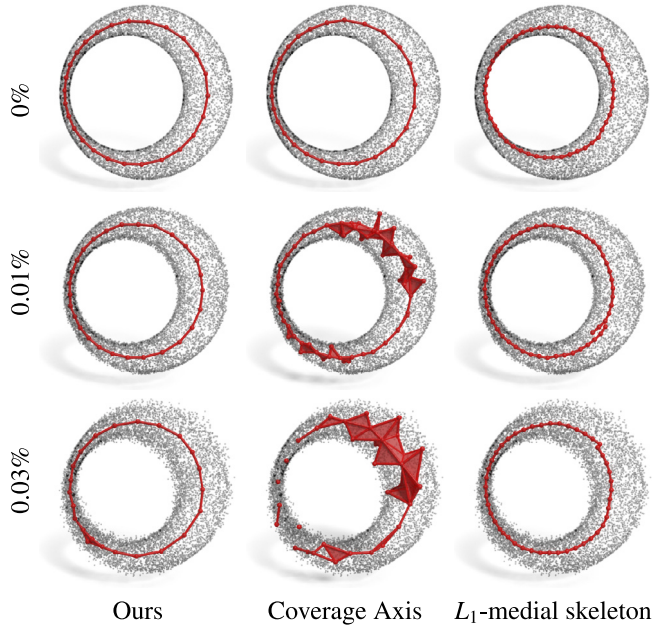


**Fig. 8.** Comparison with Coverage Axis and $L_1$-medial skeleton on a torus with different levels of noise (no noise, 0.01% and 0.03% of the shape's diagonal).

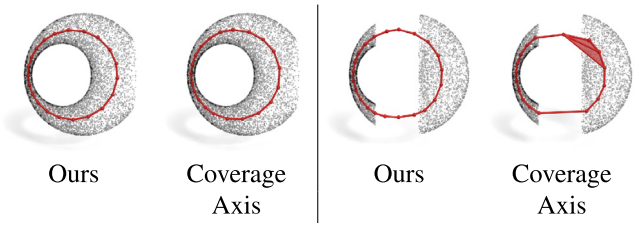


**Fig. 9.** Comparison with Coverage Axis on point clouds with missing data.

**Table 1**
Quantitative comparisons on a synthetic sphere-mesh shape, cropped or degraded with increasing noise. The Hausdorff distance is lower with our method. IGR, Mean Curvature Skeleton and VoxelCores yield worse performance. Percentage values for the noise correspond to the noise variance (percentage of the diagonal).

| Shape | Ours | SIREN | IGR | MCS | Voxel cores |
|---|---|---|---|---|---|
| clean | 0.42 | 7.9 | 1.2 | 2.4 | 0.41 |
| crop1 | 1.04 | 1.1 | 1.4 | 2.5 | 1.3 |
| crop2 | 1.9 | 2.0 | 1.5 | 2.6 | 2.0 |
| crop3 | 0.77 | 7.9 | 1.2 | 2.6 | 1.15 |
| crop4 | 0.46 | 1.5 | 2.7 | 2.5 | 1.5 |
| sub 25% | 0.35 | 8.3 | 0.86 | 2.6 | 0.42 |
| sub 50% | 0.38 | 8.1 | 1.2 | 2.5 | 0.37 |
| var 0.05% | 0.46 | 8.3 | 1.3 | 2.5 | 0.40 |
| var 0.1% | 0.45 | 7.9 | 1.1 | 2.6 | 0.39 |
| var 1% | 0.49 | 0.79 | 1.9 | 2 | 0.67 |
| var 2% | 0.57 | 0.97 | 3 | 0.84 | 1.3 |

skeleton and computing the Hausdorff distance with respect to the skeletal points. While we could have computed the distance with respect to the extracted skeleton, we found that it made the error dependent on the last mesh extraction step (MILP), which varies more and depends on a threshold $\delta$, which has to be carefully chosen for each INR. For this experiment, the input shape is built as a sphere-mesh [58], for which the exact medial axis is known by construction. We sample points on the sphere mesh surface, and apply different noises and subsamplings. In terms of Hausdorff distance our results are consistently better than the other neural baselines, and traditional methods such as VoxelCores [42] or Mean Curvature Skeleton (MCS [32]).

## 10. Ablation study

In this section we analyze several individual components of our method and compare the results on Table 2. We then ablate several components of our method: we change the activation from Sine to ReLU or SoftPlus, remove the total variation loss, remove the learning points loss, and remove the eikonal loss. Notice that with ReLU, the TV term is useless since second order derivatives are 0. In most cases, the full method yields a lower Hausdorff distance. It can be explained by the fact that TV enforces the locality of the skeleton, and no outlier skeletal points cannot lie too far away from the true skeleton. While the uniform resampling of the samples does not appear significant in Table 2 on this, for more complex shapes we found that it allowed to recover thin structures better.

We show the SDF, gradient norm field and gradient of the gradient norm for a noisy torus point set on Fig. 10. Interestingly, the 0 level set remains almost the same for the full loss (first row) or without TV regularization (second row), and the surface can be reliably extracted from both. However the gradient norm field is smooth for the full loss, whereas it is very noisy without the TV regularization. The resulting skeleton is consequently way

**Table 2**
Ablation study on a torus with added noise and cropped parts (Hausdorff distance).

| Shape | Ours | No TV | ReLU No TV | SoftPlus | SoftPlus No TV | No uniform resampling | No learning loss |
|---|---|---|---|---|---|---|---|
| Torus | 0.058 | 0.062 | 0.18 | 0.073 | 0.067 | 0.058 | 0.78 |
| Noise 0.003 | 0.011 | 0.017 | 0.24 | 0.038 | 0.038 | 0.01 | 0.79 |
| Noise 0.005 | 0.015 | 0.019 | 0.24 | 0.028 | 0.037 | 0.014 | 0.70 |
| Noise 0.01 | 0.021 | 0.18 | 0.25 | 0.035 | 0.045 | 0.021 | 0.79 |
| Noise 0.03 | 0.25 | 0.27 | 0.28 | 0.27 | 0.095 | 0.25 | 0.72 |
| Truncated 1 | 0.13 | 0.30 | 0.28 | 0.15 | 0.15 | 0.29 | 0.72 |
| Truncated 2 | 0.11 | 0.38 | 0.41 | 0.13 | 0.13 | 0.12 | 0.71 |
| Truncated 3 | 0.12 | 0.27 | 0.27 | 0.18 | 0.14 | 0.12 | 0.72 |

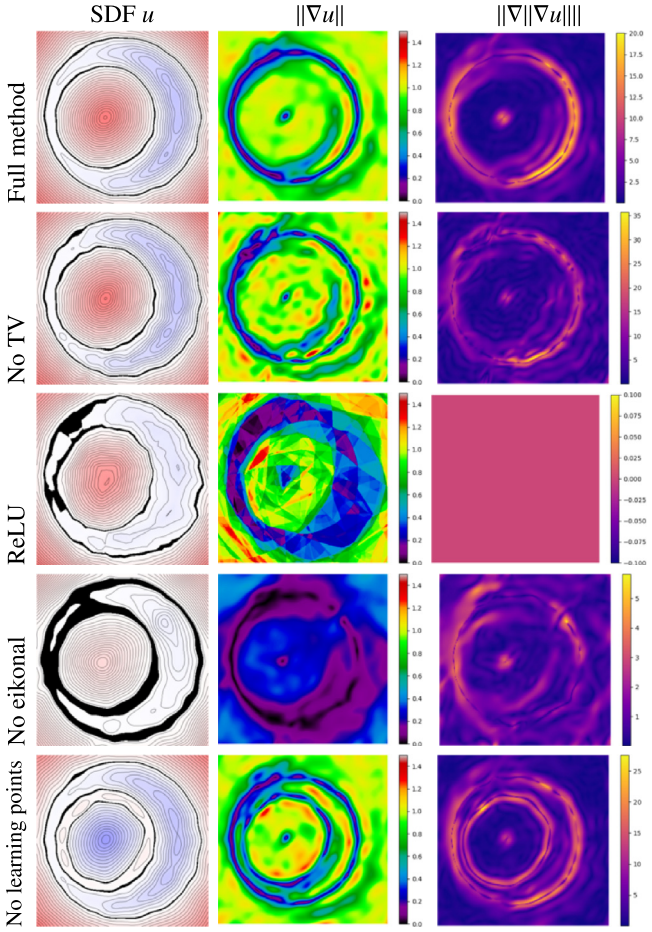


**Fig. 10.** Slices of $u$, $\|\nabla u\|$ and $\|\nabla\|\nabla u\|\|$ obtained by removing different parts of the loss.

noisier without TV. This tends to prove empirically that our initial hypothesis of adding a regularization to deal with the SDF away from a surface was the right one. Removing the eikonal loss degrades the resulting skeleton drastically, the signed distance field does no grow linearly anymore and the learning points are also critical to avoid creating wrong bubble surfaces.

Fig. 11 shows the convergence of the different terms of the Loss. Since the network is pretrained on the SDF of a sphere, the eikonal and TV losses are very low at the beginning of the training, it then increases to adapt the input shape before decreasing again. The losses converge in a rather smooth way.

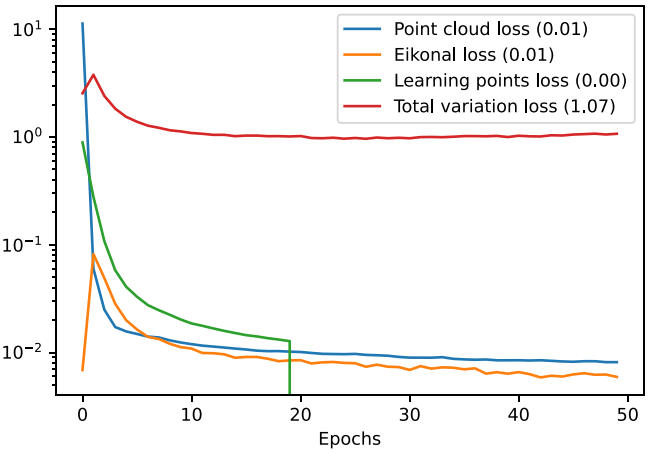


**Fig. 11.** Evolution of the different terms of the loss. After 20 epochs, we remove the learning points loss (Eq. (5)) to allow for more flexibility since in case of noise or missing data, the learning points might be inaccurate.

The computation time is divided as follows: INR training takes between 55 s and 65 s (around 45 s for SIREN), the shape sampling and skeletal extraction step takes 1–2 s. The coverage axis step is more unpredictable with respect to computation time. The optimal solution can be found in 30 s to 60 s for simple shapes, but it can take longer to reach optimality (up to 1200 s) for more complex shapes such as the birdcage of Fig. 1. Fortunately we can set a time-out at 1000 s and check by looking at the optimal bounds if the remaining gap is large or not. In general after 1000 s, the gap is small and only a few vertices (around 4–10) could be further removed.

*Limitations.* A limitation of our neural implicit formulation remains the need for a normal, and more importantly consistently oriented normals which are hard to obtain for complex shapes. We believe this could be improved by working on unsigned distance fields [15], possibly yielding two skeletons, one inside the shape and one outside. However this would also require rethinking the skeleton extraction step, since both the surface and the medial axis are singularities of the unsigned distance gradient.

## 11. Conclusion

In this paper we presented a new TV-regularized neural implicit, which allows to perform computations away from the surface. While we introduced it in conjunction with a sine-activation network, hence gaining a stable differentiation of the implicit field, the regularization could be extended to other networks, including those that work on latent shape space such as DeepSDF.

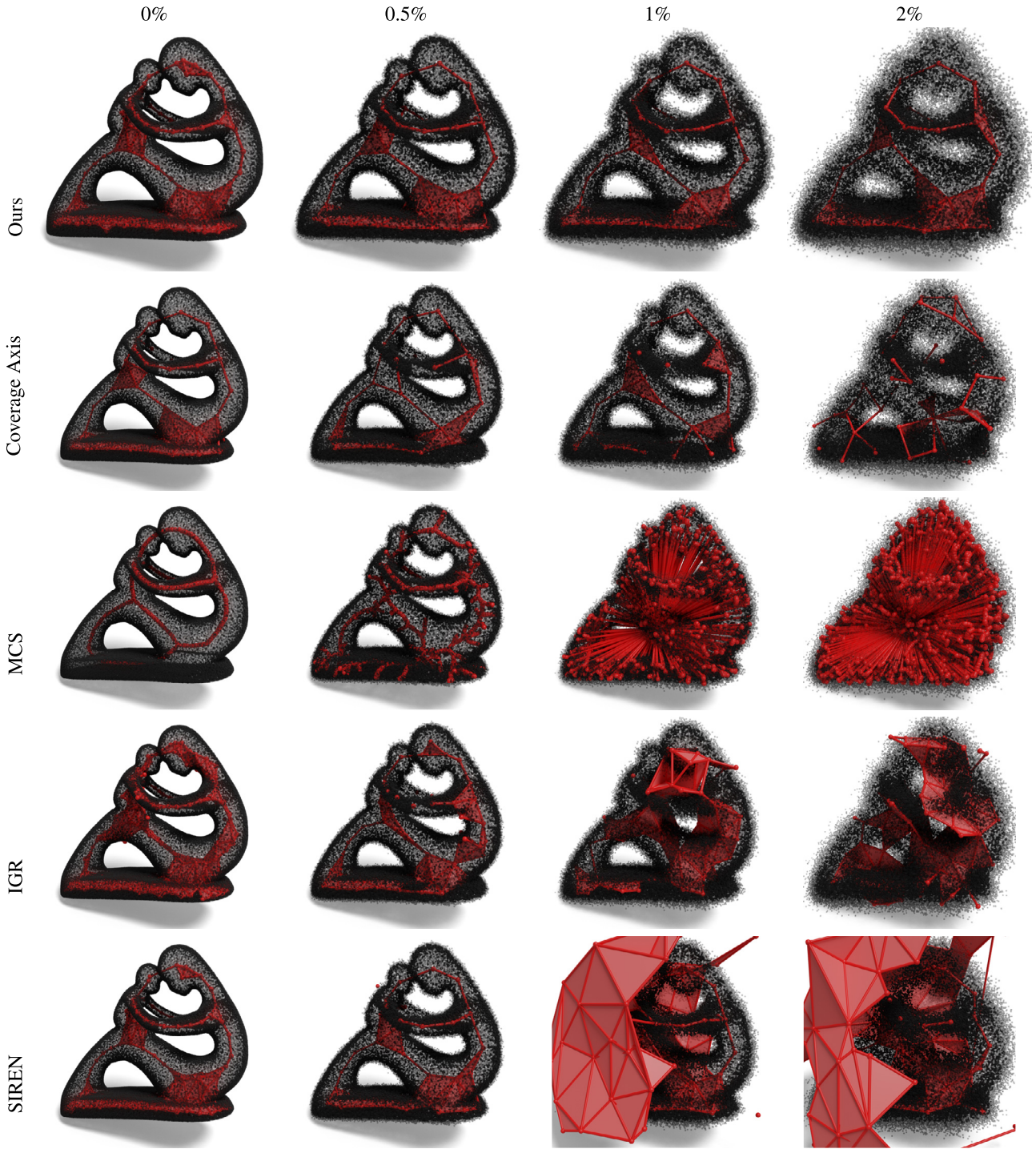


**Fig. 12.** Skeleton of the fertility shape with increasing levels of noise (variances set to 0, 0.5, 1, and 2% of the shape diagonal).

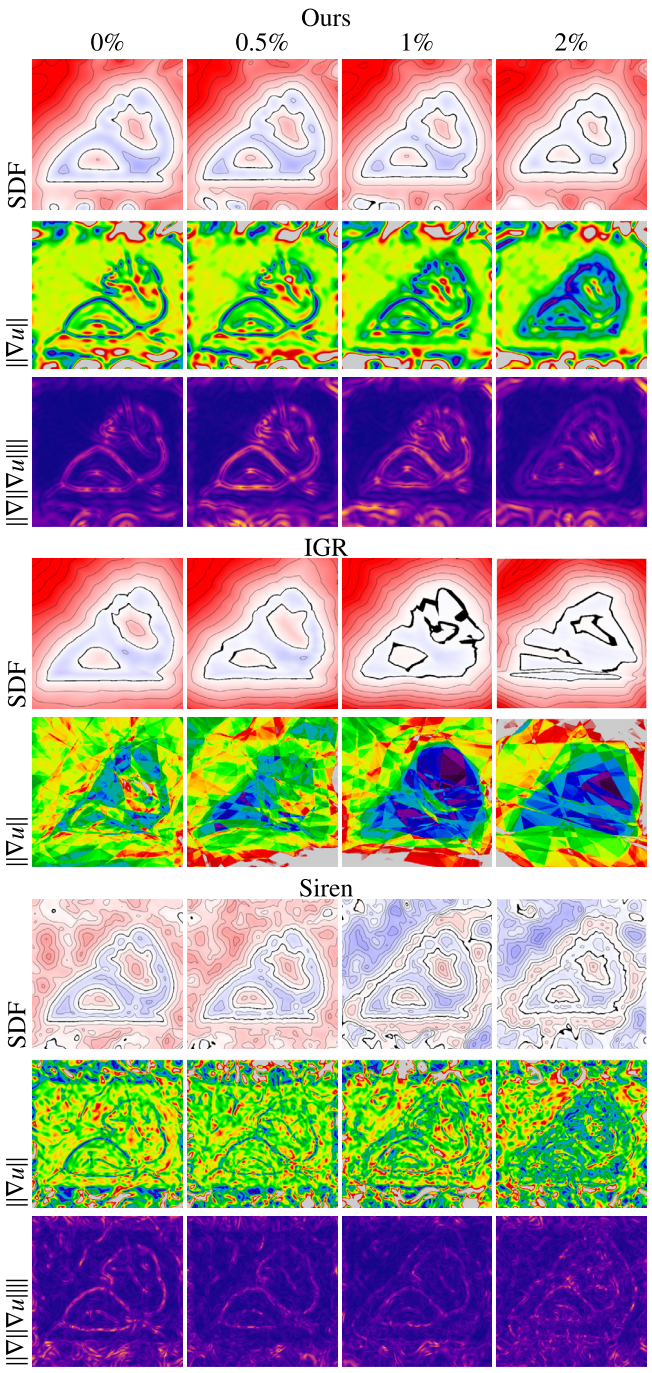


**Fig. 13.** Slices of the SDF and the gradient's norm of the fertility shape with increasing levels of noise for different INR (variances set to 0, 0.5, 1, and 2% of the shape diagonal). Since IGR's second order derivatives are 0, we do not show $\|\nabla\|\nabla u\|\|$ for this method.

## CRediT authorship contribution statement

**Mattéo Clémot:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – original Draft , Writing – review & editing, Formal analysis. **Julie Digne:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – original Draft , Writing – review & editing, Formal analysis, Visualization, Supervision, Project administration, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The data and code will be made available through github upon acceptance and submitted to GRSI.

## Acknowledgments

This work was supported by the French Agence Nationale de la Recherche (ANR), under grant ANR-19-CE45-0015 (ToPACS). We thank Hsueh-Ti Derek Liu for providing the Blender Toolbox which was extensively used in this paper.

## References

[1] Xie Y, Takikawa T, Saito S, Litany O, Yan S, Khan N, et al. Neural fields in visual computing and beyond. Comput Graph Forum 2022.
[2] Bloomenthal J, Wyvill B. Interactive techniques for implicit modeling. In: Proceedings of I3D 1990. 1990, p. 109–16.
[3] Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Surface reconstruction from unorganized points. SIGGRAPH Comput Graph 1992;26(2):71–8.
[4] Lorensen WE, Cline HE. Marching cubes: A high resolution 3D surface construction algorithm. In: SIGGRAPH. ACM; 1987.
[5] Hart JC. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. Vis Comput 1996;12:527–45.
[6] Park JJ, Florence P, Straub J, Newcombe R, Lovegrove S. DeepSDF: Learning continuous signed distance functions for shape representation. In: CVPR 2019. 2019.
[7] Mescheder L, Oechsle M, Niemeyer M, Nowozin S, Geiger A. Occupancy networks: Learning 3D reconstruction in function space. In: CVPR. 2019.
[8] Rudin LI, Osher S, Fatemi E. Nonlinear total variation based noise removal algorithms. Physica D 1992;60(1):259–68.
[9] Qi CR, Su H, Mo K, Guibas LJ. PointNet: Deep learning on point sets for 3D classification and segmentation. In: IEEE CVPR. 2017.
[10] Hanocka R, Hertz A, Fish N, Giryes R, Fleishman S, Cohen-Or D. MeshCNN: A network with an edge. ACM Trans Graph 2019;38(4).
[11] Chen Z, Zhang H. Learning implicit fields for generative shape modeling. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2019.
[12] Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R. NeRF: Representing scenes as neural radiance fields for view synthesis. In: ECCV. 2020.
[13] Atzmon M, Lipman Y. SAL: Sign agnostic learning of shapes from raw data. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.
[14] Atzmon M, Lipman Y. SALD: sign agnostic learning with derivatives. In: ICLR 2021. 2021.
[15] Chibane J, Mir A, Pons-Moll G. Neural unsigned distance fields for implicit function learning. In: Advances in Neural Information Processing Systems. 2020.
[16] Gropp A, Yariv L, Haim N, Atzmon M, Lipman Y. Implicit geometric regularization for learning shapes. In: Proceedings of Machine Learning and Systems 2020. 2020.
[17] Peng S, Niemeyer M, Mescheder L, Pollefeys M, Geiger A. Convolutional occupancy networks. In: Computer Vision – ECCV 2020. 2020, p. 523–40.
[18] Chabra R, Lenssen JE, Ilg E, Schmidt T, Straub J, Lovegrove S, et al. Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In: ECCV 2020. 2020, p. 608–25.
[19] Jiang CM, Sud A, Makadia A, Huang J, Nießner M, Funkhouser T. Local implicit grid representations for 3D scenes. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition. 2020.
[20] Sitzmann V, Martel JN, Bergman AW, Lindell DB, Wetzstein G. Implicit neural representations with periodic activation functions. In: NeurIPS. 2020.
[21] Yifan W, Rahmann L, Sorkine-hornung O. Geometry-consistent neural shape representation with implicit displacement fields. In: International Conference on Learning Representations. 2022.
[22] Takikawa T, Litalien J, Yin K, Kreis K, Loop C, Nowrouzezahrai D, et al. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. 2021, arXiv.

[23] Liu H-TD, Williams F, Jacobson A, Fidler S, Litany O. Learning smooth neural functions via Lipschitz regularization. 2022, arXiv.
[24] Lipman Y. Phase transitions, distance functions, and implicit neural representations. 2021, CoRR. arXiv:2106.07689.
[25] Williams F, Trager M, Bruna J, Zorin D. Neural splines: Fitting 3D surfaces with infinitely-wide neural networks. In: CVPR 2021. 2021.
[26] Müller T, Evans A, Schied C, Keller A. Instant neural graphics primitives with a multiresolution hash encoding. TOG Siggraph 2022;41(4).
[27] Sharp N, Jacobson A. Spelunking the deep: guaranteed queries on general neural implicit surfaces via range analysis. ACM Trans Graph 2022;41(4).
[28] Tagliasacchi A, Delame T, Spagnuolo M, Amenta N, Telea A. 3D skeletons: A state-of-the-art report. Comput Graph Forum 2016;35(2).
[29] Dey TK, Sun J. Defining and computing curve-skeletons with medial geodesic function. In: SGP 2006. Eurographics; 2006, p. 143–52.
[30] Au OK-C, Tai C-L, Chu H-K, Cohen-Or D, Lee T-Y. Skeleton extraction by mesh contraction. ACM Trans Graph 2008;27(3):1–10.
[31] Livesu M, Guggeri F, Scateni R. Reconstructing the curve-skeletons of 3D shapes using the visual hull. IEEE Trans Vis Comput Graphics 2012;18(11):1891–901.
[32] Tagliasacchi A, Alhashim I, Olson M, Zhang H. Mean curvature skeletons. In: Computer Graphics Forum. 2012.
[33] Tagliasacchi A, Zhang H, Cohen-Or D. Curve skeleton extraction from incomplete point cloud. In: ACM Transactions on Graphics. 2009.
[34] Huang H, Wu S, Cohen-Or D, Gong M, Zhang H, Li G, et al. L1-medial skeleton of point cloud. ACM Trans Graph 2013;32(4).
[35] Blum H. A transformation for extracting new descriptors of shape. In: Models for Perception of Speech and Visual Form. Cambridge, MA: MIT Press; 1967.
[36] Attali D, Boissonnat J-D, Edelsbrunner H. Stability and computation of medial axes: a state-of-the-art report. In: Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration. Mathematics and visualization, Springer-Verlag; 2009, p. 109–25.
[37] Amenta N, Choi S, Kolluri RK. The power crust, unions of balls, and the medial axis transform. Comput Geom Theory Appl 2001;19(2–3):127–53.
[38] Dey TK, Zhao W. Approximate medial axis as a voronoi subcomplex. In: Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications. Association for Computing Machinery; 2002, p. 356–66.
[39] Li P, Wang B, Sun F, Guo X, Zhang C, Wang W. Q-MAT: Computing medial axis transform by quadratic error minimization. ACM Trans Graph 2016;35(1).
[40] Rebain D, Angles B, Valentin J, Vining N, Peethambaran J, Izadi S, et al. LSMAT least squares medial axis transform. Comput Graph Forum 2019;38(6):5–18.
[41] Wu S, Huang H, Gong M, Zwicker M, Cohen-Or D. Deep points consolidation. ACM Trans Graph 2015;34(6).
[42] Yan Y, Letscher D, Ju T. Voxel cores: Efficient, robust, and provably good approximation of 3D medial axes. ACM Trans Graph 2018;37(4).
[43] Dou Z, Lin C, Xu R, Yang L, Xin S, Komura T, et al. Coverage axis: Inner point selection for 3D shape skeletonization. In: Computer Graphics Forum - Eurographics 2022. 2022.
[44] Xu Z, Zhou Y, Kalogerakis E, Singh K. Predicting animation skeletons for 3D articulated models via volumetric nets. In: 2019 International Conference on 3D Vision. 2019.
[45] Lin C, Li C, Liu Y, Chen N, Choi Y-K, Wang W. Point2Skeleton: Learning skeletal representations from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, p. 4277–86.
[46] Yang B, Yao J, Wang B, Hu J, Pan Y, Pan T, et al. P2MAT-NET: Learning medial axis transform from sparse point clouds. Comput Aided Geom Design 2020;80.
[47] Hu J, Wang B, Qian L, Pan Y, Guo X, Liu L, et al. MAT-net: Medial axis transform network for 3D object recognition. In: IJCAI 2019. 2019.
[48] Rebain D, Li K, Sitzmann V, Yazdani S, Yi KM, Tagliasacchi A. Deep medial fields. 2021, CoRR. arXiv:2106.03804.
[49] Osher S, Sethian JA. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. J Comput Phys 1988;79(1):12–49.
[50] Barles G. An introduction to the theory of viscosity solutions for first-order Hamilton–Jacobi equations and applications. In: Hamilton-Jacobi Equations: Approximations, Numerical Analysis and Applications. Springer; 2013, p. 49–109.
[51] Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Math Program 1989;45(1):503–28.
[52] Yifan W, Wu S, Öztireli C, Sorkine-Hornung O. Iso-points: Optimizing neural implicit surfaces with hybrid representations. In: CVPR. 2021.
[53] Rouvreau V. Alpha complex. In: GUDHI User and Reference Manual. GUDHI Editorial Board; 2015, URL https://gudhi.inria.fr/doc/latest/group__alpha__complex.html.
[54] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental algorithms for scientific computing in python. Nature Methods 2020;17:261–72.
[55] Zhou Q, Jacobson A. Thingi10K: A dataset of 10,000 3D-printing models. 2016, arXiv.
[56] Osada R, Funkhouser T, Chazelle B, Dobkin D. Shape distributions. ACM Trans Graph 2002;21(4):807–32.
[57] Ju T. Robust repair of polygonal models. ACM Trans Graph 2004;23(3).
[58] Thiery J-M, Guy É, Boubekeur T, Eisemann E. Animated mesh approximation with sphere-meshes. ACM Trans Graph 2016;35(3).