Pacific Graphics 2025
M. Christie, N. Pietroni, and Y.-S. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 7

Procedural Multiscale Geometry Modeling using Implicit Surfaces (supplemental)

Bojja Venu[1], Adam Bosak[1] and Juan Raúl Padrón-Griffe[2]

[1] Technical University of Denmark, Denmark
[2]Universidad de Zaragoza-I3A, Spain

This document covers additional details of the modeling of the multiscale geometry in Sec. 1, the optimization of the sphere tracing algorithm for visualization of the multiscale geometry in Sec. 2, and the reconstruction of the multiscale geometry in Sec. 3.

1. Multiscale geometry modeling

This section covers additional details and analysis of multiscale geometry modeling.

1.1. Tiling and Issues

Tiling introduces regularity artifacts between tiles, and continuous spatial variations using the tiling approach also introduce artifacts and breaks between the tiles. In addition, density variations using tiling produce inconsistent shading. In Fig. 1 we illustrate the tiling artifacts.

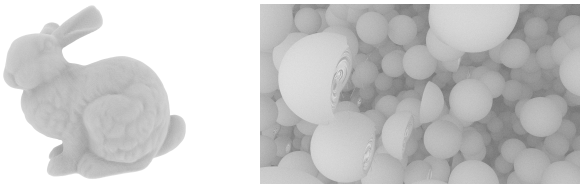

Figure 1: *Tiling artifacts. Tiling introduces regularity artifacts and generates errors between tiles, making it impossible to create continuous spatial variations.*

1.2. Point Distribution

The following is a way we can generate the seed for the random number generator and add the random vector to the particle in the grid cell $\boldsymbol{q}$ to create a random distribution in the eight neighbors (adding (0,0,0), (1,0,0), (0,1,0), (1,1,0), (0,0,1), (1,0,1), (0,1,1), (1,1,1) to the point $\boldsymbol{p}$ to get the grid cell $\boldsymbol{q}$) of the point $\boldsymbol{q}$.

$$t = N\sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} a_{ijk}q_x^i q_y^j q_z^k \tag{1}$$

$$(\xi_1,\ldots,\xi_N) = \mathrm{rnd}(t,N) \tag{2}$$

$$[\boldsymbol{x},s] = [\boldsymbol{q}+(\xi_1,\xi_2,\xi_3),\xi_4], \tag{3}$$

The pseudorandom number generator (PRNG), denoted as rnd, produces N random numbers that are uniformly distributed within the interval $[0,1)$. The coefficients a_{ijk} are chosen to remove regularity artifacts in the distribution of particles that result in the process of generating particle clouds. Choosing prime numbers as coefficients eliminates the regularity artifacts in a better way.

1.3. Multi-phase Particle Cloud

We use spatial transformations of points to generate multiphase particles within a volume. The following transformation generates particles with various shapes, smoothly transitioning between each type of shape in the volume, where each shape is considered a different phase. This works as a frequency modulation of the signal, and it almost generates the Phasor noise [TEZ*19] effect. These transformed points are used to generate the particle cloud.

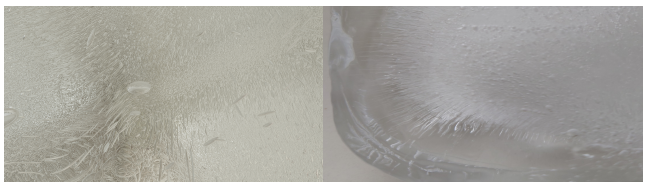

Figure 2: *Inspired by a photo of air bubbles in ice (right), we used our multi-phase particle cloud approach to model a similar material (left). Our model has ice as the host medium and contains air particles that vary in size and shape with the spatial location in the medium, transitioning from spherical to non-spherical.*

We can create a particle cloud containing specific regions where some regions are isotropic and others are anisotropic, with smooth transitions between these regions, see Fig. 2. This concept is similar to positional encoding, where particular regions in the volume behave in a specific manner. To achieve this, the transformed point

$$\boldsymbol{p'} = \boldsymbol{pT}, \quad \text{with } \boldsymbol{T} = \boldsymbol{AD}, \tag{4}$$

is used as the input point for the particulate material generation. This generates effects like phasor noise, where the signal's frequency

depends on spatial coordinates. The matrices

$$\boldsymbol{A} = \begin{bmatrix} A_1 & A_2 & A_3 \\ A_4 & A_5 & A_6 \\ A_7 & A_8 & A_9 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} \sin(p_y) & \sin(p_z) & \sin(p_x) \\ \sin(p_z) & \sin(p_x) & \sin(p_y) \\ \sin(p_x) & \sin(p_y) & \sin(p_z) \end{bmatrix}, \tag{5}$$

contribute to these phasor noise generation effects. The matrix $\boldsymbol{A}$ acts as an amplitude matrix, typically with values between 0 and 1. Essentially, we use the original point to generate a noise matrix or turbulence [PH89] matrix ($\boldsymbol{T}$), which is a circulant-type matrix that is symmetric and every row is a left circular shift of the previous row.

1.4. Mesoscale Surface Patterns on the Macrosurface

The following distance approximations can be used to generate mesoscale surface patterns on macroscopic objects with thickness control.

$$d(\boldsymbol{p}, w) = \begin{cases} d_g(\boldsymbol{p}, w), & \text{if } \exists \boldsymbol{q}_k \text{ such that } l_i < P_i(\boldsymbol{q}_k) < r_i, \\ & \forall k = 0, \ldots, 26, \\ \frac{w}{2}, & \text{otherwise.} \end{cases} \tag{6}$$

where,

$$d_g(\boldsymbol{p}, w) = f(\boldsymbol{p}) + \min_{\substack{k=0 \\ \exists \boldsymbol{q}_k,\, l_i < P_i(\boldsymbol{q}_k) < r_i}}^{26} d_{\boldsymbol{q}_k}\left(\frac{1}{w}(\boldsymbol{p} + \boldsymbol{h}(\boldsymbol{p}))\right), \tag{7}$$

In this condition, the terms l_i and r_i represent the minimum and maximum bounds of the corresponding polynomial values $P_i(\boldsymbol{q_k})$ when the i-th polynomial is evaluated with the point $\boldsymbol{q_k}$. The interval length (l_i, r_i) controls the width (or thickness) of the macroscopic object. In some cases, it may be necessary to use R instead of P. These polynomials can be used to create the macro-shape of the object based on the roots of the polynomials; basically, the particle jittering follows these polynomial functions. When the roots of these polynomials are true, the microgeometry on the surface can be generated using the distance bound $d_g(\boldsymbol{p}, w)$.

For instance, the mesoscale geometry on the macrosurface is shown in Fig. 3. The following polynomial can then be used to generate the macrosurface with microstructure.

$$P(\boldsymbol{q}) = 2q_x + q_y^2 + q_z^2 - q_x q_y - q_y q_z - q_z q_x. \tag{8}$$

1.5. Implicit Periodic Functions

Implicit periodic functions, such as sine and cosine, can avoid grid discretizations and improve performance, as they do not require geometry evaluation in neighboring grid cells. Many geometric structures, such as fibers, can be represented by combinations of these functions with varying frequencies and amplitudes following the Fourier Series. In addition, periodic functions are more suitable for continuous optimization.

The general formula which can be used to generate various structures with sine and cosine functions defined is

$$SC(\boldsymbol{p}) = \sum_{i=1}^{u} \prod_{j=1}^{v} T_{ij}(\boldsymbol{p})^{k_{ij}} + w, \tag{9}$$

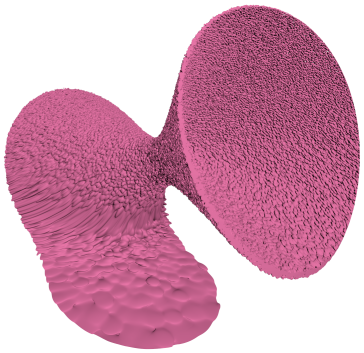

Figure 3: *Mesoscale texture patterns on the macrosurface, where mesoscale patterns follow the macrosurface geometry. The same Implicit function can generate both the mesoscale geometry and the second-order macro surface, with control over thickness through particle agglomeration. As a result, we can see mesoscale geometry at the surface level.*

where w represents the width of the microstructure and $k_{ij} \geq 0$ are the powers associated with each trigonometric term $T_{ij}(\boldsymbol{p})$ defined by

$$T_{ij}(\boldsymbol{p}) \in \{A_k \cdot f(\omega_k p_d + \phi_k) \mid f \in \{\sin, \cos\},\, p_d \in \{p_x, p_y, p_z\}\}.$$

where A_k is the amplitude, ω_k is the frequency, and ϕ_k is the phase shift of the trigonometric function.

One collection of such functions that we use in our system is the family of Triply Periodic Minimal Surfaces (TPMS); for instance, the Gyroid, Diamond, and Primitive functions [HC18], see Sec. 3.1 for equations of these structures.

Many microstructures can be defined using these trigonometric functions: Let us consider a 3×3 matrix $\boldsymbol{T}$, which performs affine transformations such as scaling, rotation, and translation. Sometimes, one or more of these operations are combined in the 3×3 matrix using octaves. Each entry in this matrix is generated from the summation of multiple sine and cosine waves with varying amplitudes and frequencies, and typically, each value is normalized between -1 and 1, see Sec. 3.1.4.

1.6. Particle Piling

In this section, we report additional experiments for piling structures with our framework. The implicit surfaces for the particle material can be changed in the following manner for the generation of particle piling:

$$d_g(\boldsymbol{p}, w) = f(\boldsymbol{p}) + \min_{i=0,\ldots,26} d_{\boldsymbol{q}_i}\left(\frac{1}{w}((\boldsymbol{R}_\theta \boldsymbol{p}) + \boldsymbol{P}(\boldsymbol{p})\boldsymbol{N}(\boldsymbol{p}))\right), \tag{10}$$

where $f(p)$ is an arbitrary function for spatially varying surface offsetting. The term $\boldsymbol{R}_\theta$ refers to the rotation matrix with angle θ. The term $\boldsymbol{P}(\boldsymbol{p})\boldsymbol{N}(\boldsymbol{p})$ generates the implicit surface deformation using a product of the control polynomial function $P(\boldsymbol{p})$ and the noise function $N(\boldsymbol{p})$. The term $P(\boldsymbol{p})$ in the above equation controls the degree of convexity to the concavity of the grains in the same

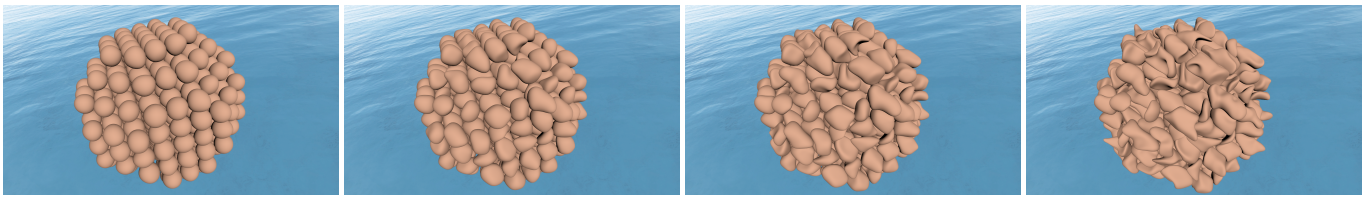

Figure 4: *Starting from a regular cubic lattice of spherical grains (first column), slowly adding different levels of sparse convolution noise [FW07] as $N(\boldsymbol{p})$ in the eq. 14 in the main document for the implicit surface deformation generates grains similar to rock piles with contact points. Increasing the level of deformation generates more concave grains. Here, the rotation angle with respect to the y-axis used is 15°. We can utilize any deformation function that deforms the implicit surface in a specific manner for targeted applications.*

Figure 5: *Our framework can generate multiscale granular media with grains in settled positions. In the left image, the generated rockpile shows smaller grains at the bottom and larger ones at the top, with well-defined contact points even across scales. The right image illustrates the same cloud, but with an additional polynomial function applied to the rotation angle, $\theta = 15 + p_z$, as defined in Eq. 14 of the main document.*

particle cloud, where the smooth transition between convexity and concavity happens in the same cloud.

In Fig. 4, we show how different levels of deformation affect an initial particle cloud of spheres. This approach is useful for modeling granular media, such as rock piles, as can be seen in the last column. Fig. 5 illustrates the versatility of our framework to generate multiscale granular patterns on the fly without the need for any pre-computation or collision detection methods.

2. Multiscale Geometry Rendering

In this section, we provide additional details about Lipschitz bounds 2.1 for the transformed objects and the multiscale sphere tracing algorithm 2.2.

2.1. Lipschitz Bounds for the Transformed Objects

The Lipschitz constant of the sum of two functions is, at most, the sum of their Lipschitz constants. The Lipschitz constant of the composite function is the product of the Lipschitz constant of each of the component function's Lipschitz constants [Har96].

The transformations, rotations, translations, and reflections are called isometries, and if the transformation $\boldsymbol{T}$ is an isometry, then the distances for the implicit function need no adjustment, meaning the distances are preserved when we perform these isometric transformations. For example, in our case, the warping function $h(\boldsymbol{p})$ is an isometry transformation, essentially a translation. Therefore, when we use something for $h(\boldsymbol{p})$ like Perlin noise in Eq. 3 in the main document, we do not need to change the Lipschitz bounds [Har96].

Another important transformation we use is the scaling of the point coordinates to achieve anisotropy in shape. Suppose we use the following scaling operation on point coordinates [Har96]:

$$\boldsymbol{p'} = (c_1 p_x, c_2 p_y, c_3 p_z)\,. \tag{11}$$

The value of the Lipschitz bound is then $\max(c_1, c_2, c_3)$.

Linear deformation of shapes is achieved by multiplying the transformation matrix $\boldsymbol{T}$ with the point $\boldsymbol{p}$, resulting in spatially varying shapes. In Sec. 1.3, the transformed point $\boldsymbol{p'}$ is obtained using this type of transformation. The value of the Lipschitz bound for this case is the largest eigenvalue of the matrix $\boldsymbol{T}$ [Har96], and we can find this using the power method [Ger04].

For example, we used the turbulence matrix $\boldsymbol{T}$ for the linear deformation in Sec. 1.3. This turbulence matrix is dependent on the point $\boldsymbol{p}$ each time. For the Lipschitz bound, we need to calculate the maximum eigenvalue of this matrix each time, which is computationally expensive. Instead of calculating it every time for matrices that change, we can calculate it only once for the matrix with all entries set to the maximum possible values. In this case, it is the value one for all entries in the matrix. The maximum eigenvalue of this matrix, using the power method, is 3. This value serves as the Lipschitz bound for the transformation involving $\boldsymbol{T}$.

2.2. Multiscale Sphere Tracing

In this paper, we present an optimized sphere-tracing algorithm aimed at improving the performance of sphere tracing for multiscale grid variations as illustrated in Alg. 1 and Fig. 6.

Algorithm 1 Optimized sphere tracing with adaptive step factors based on gradients of the implicit surface and the polynomial function $D(\boldsymbol{p})$ for sparse sampling. IS denotes the implicit surface.

1: **function** COMPUTEGRIDSCALE($\boldsymbol{p}$)
2: $\varepsilon \leftarrow 0.01$
3: $dp_x \leftarrow (\varepsilon, 0, 0)$
4: $dp_y \leftarrow (0, \varepsilon, 0)$
5: $dp_z \leftarrow (0, 0, \varepsilon)$
6: $gp_x \leftarrow |\text{IS}(p + dp_x) - \text{IS}(p - dp_x)|$
7: $gp_y \leftarrow |\text{IS}(p + dp_y) - \text{IS}(p - dp_y)|$
8: $gp_z \leftarrow |\text{IS}(p + dp_z) - \text{IS}(p - dp_z)|$
9: $|\nabla d| \leftarrow \sqrt{gp_x^2 + gp_y^2 + gp_z^2}$
10: $scale \leftarrow \text{clamp}(|\nabla d| \cdot 0.5, 0.0, 1.0)$
11: **return** *scale*
12: **end function**
13: **function** MULTISCALE-SPHERE-TRACING(ro, rd, $t_{\min}$, $t_{\max}$)
14: $t \leftarrow t_{\min}$
15: $d \leftarrow \text{IS}(ro + t \cdot rd)$
16: $s \leftarrow \text{sign}(d)$
17: $lastScale \leftarrow \text{ComputeGridScale}(ro + t \cdot rd)$
18: **for** $i = 0$ to n **do**
19: **if** $|d| < \text{precision} \cdot t$ **or** $t > t_{\max}$ **then**
20: **break**
21: **end if**
22: **if** $D(\boldsymbol{p}) = 1$ **then** ▷ Evaluate gradient only when polynomial checks $D(\boldsymbol{p})$ is true
23: $lastScale \leftarrow \text{ComputeGridScale}(ro + t \cdot rd)$
24: **end if**
25: $stepFactor \leftarrow \text{LERP}(\delta_{\min}, 1.0, lastScale)$
26: $t \leftarrow t + s \cdot d \cdot stepFactor$
27: $d \leftarrow \text{IS}(ro + t \cdot rd)$
28: **end for**
29: **return** t
30: **end function**

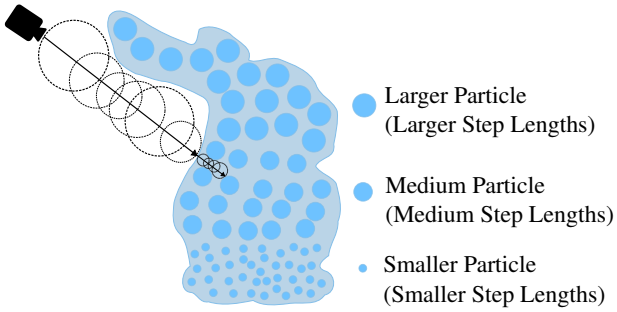

Figure 6: *We improve the efficiency of the sphere tracing algorithm using adaptive step lengths controlled by the magnitude of the implicit surface gradients. Polynomial checks allow sparse sampling of gradients, reducing the number of expensive gradient evaluations while still capturing grid-scale variations. This enables finer grids to use shorter steps and coarser grids to use longer steps, improving overall performance. An example of a polynomial condition is* $\sin(p_x)\cos(p_y) + \sin(p_y)\cos(p_z) + \sin(p_z)\cos(p_x) = 0.5$.

In the above algorithm, on line 22, we can use any other condition to reduce the gradient computations and improve performance during sphere-tracing steps. For instance, we can check this using the following condition, where we examine gradients to track variations in grid scale for every m-step length.

$$\text{If } i \mod m = 0, \text{ where } m < n$$

For instance, we use something like the following functions (in Eqs. 12 and 13) for the $D(\boldsymbol{p})$ in the above algorithm to track the gradient changes. Essentially, we are sampling sparse points throughout the volume to track the grid scale variations effectively. The polynomial conditional check $D(\boldsymbol{p})$ is a method to reduce the number of gradient evaluations in sphere tracing iterations.

$$D(\boldsymbol{p}) = \begin{cases} 1, & \text{if } |\text{fmod}(p_y, n_1)| = 0.0 \\ & \vee\,(|\text{fmod}(p_z, n_2)| = 0.0 \wedge |\text{fmod}(p_x, n_3)| = 0.0) \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

Step Factor	RTX 4090		RTX A2000	
	RT (ms)	FPS	RT (ms)	FPS
Fixed global bound	9.3	105.7	50	20
Bisection based	9.0	108.5	45	22.1
Fully Adaptive (N = 1)	44.1	22.6	240	4
Adaptive (N = 10)	11.5	85.8	60.6	16.2
Adaptive (N = 100)	8.1	121.1	44	22.8
Adaptive (N = 500)	8.1	121.5	42.7	23.1
Adaptive ($D(\boldsymbol{p}) = 1$)	7.8	124.5	42.2	23.5

Table 1: *Performance comparison of different step-length strategies for sphere tracing in multiscale particulate media made of Lambertian particles. We evaluate fixed stepping (classical sphere tracing), the bisection method, fully adaptive stepping (update every step), adaptive stepping with gradient checks every N steps (10,100 and 500), and adaptive stepping with polynomial-based gradient checks for 1500 steps. Longer steps are more effective for macroscopic scales, shorter steps for microscopic scales, and intermediate steps for mesoscopic scales. Adaptive stepping with polynomial checks achieves the best performance, improving rendering efficiency by 16% over fixed and bisection methods..*

$$D(\boldsymbol{p}) = \begin{cases} 1, & \text{if } \sin(p_x)\cos(p_y) + \sin(p_y)\cos(p_z) \\ & \quad + \sin(p_z)\cos(p_x) = 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

These methods can improve performance, but using polynomial checks to track variations in the grid scale is particularly effective. It allows smaller step lengths in finer grid-scale regions and larger ones in coarser grid scales. In Table 1, we report the performance comparison of different sampling strategies for the step length across multiple GPUs. Fig. 7 presents the renderings obtained with the multiscale sphere tracing algorithm, demonstrating that the use of polynomial checks yields a significant performance boost without introducing noticeable visual differences.

Figure 7: *Visualization of the rendered multiscale grid variational geometry with different step length methods. The results are shown for fixed step lengths, as well as for the two polynomial checks $D(\boldsymbol{p})$ (as referenced in equations 12 and 13) used for adaptive step lengths, respectively. The performance values for these three cases are as follows: for the fixed lengths case, the results are (RT - 8.6 ms, FPS - 112.3). For the polynomial-based gradient checks with the equation in 12, where the values of $n1, n2, n3$ are 11, 5, and 7, the results are (RT - 7.0 ms, FPS - 140.1). With the equation in 13, the results are (RT - 6.9 ms, FPS - 141.6). It clearly shows that polynomial-based gradient checks for tracking grid scale variations within the sphere tracing algorithm improve performance.*

3. Multiscale Geometry Reconstruction

Inverse procedural modeling promises to overcome some of the main limitations of procedural modeling, such as parameter tweaking to create a given real target exemplar. In this section, we focus on reconstructing procedural microstructures generated by our framework. We explore two modalities: implicit surfaces, potentially extracted from noisy MicroCT scans, and RGB images, potentially obtained from Transmission Electron Microscopy (TEM) or Scanning Electron Microscopy (SEM) samples. First, we create a synthetic dataset of representative microstructures that serves as a proof of concept. For the first modality, we apply direct parameter fitting to find the parameters of our procedural microstructures with observed data. For the second modality, we employ analysis-by-synthesis, iteratively refining reconstructions by comparing a simulated image with a reference image. We also show that the reconstruction of synthetic microstructures is not a trivial task by trying to reconstruct implicit functions using popular network architectures to represent signals such as NeRF [MST*21] and SIREN [SMB*20], and a physically-based differentiable rendering algorithm for SDF [VSJ22].

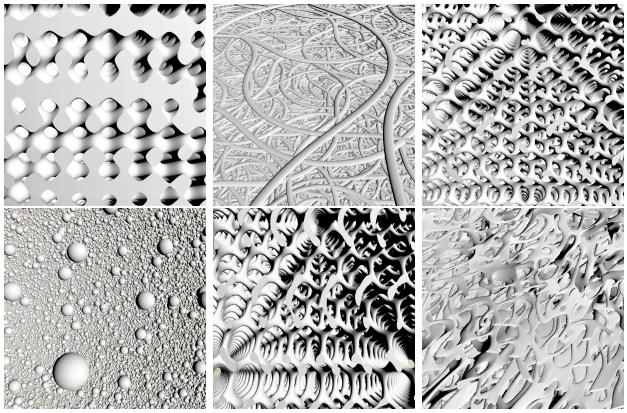

Figure 8: *Instances of exemplar renders (top row) Gyroid [1D] 14, Fibers [2D] 25, Gyroid [3D] 15 respectively. Rest of the renders (bottom row) are Spheres [2D] 22, Gyroid [5D] 18, Porous [28D] 28 respectively. The number inside the parentheses corresponds to the number of parameters, i.e., the dimensionality of the space search.*

3.1. Synthetic Dataset

For the proof of concept and more control over the experiments, we evaluate our algorithms on representative synthetic examples that resemble SEM/TEM images. The dataset consists of six microstructures, where each microstructure is defined by an implicit function $\hat{f}$ with n parameters, denoted as $\boldsymbol{\phi}$. All ground-truth parameter values presented in the dataset were selected for their convenience and to ensure they fit in the defined domain range.

3.1.1. Gyroid Family

The first family of microstructures are the gyroid microstructures. These consist of periodic fully deterministic patterns created using trigonometric functions. Our dataset includes three such microstructures with 1, 3, and 5 degrees of freedom, respectively. The simplest example is the Gyroid [1D]. The only parameter is a scalar variable $\eta \in \mathbb{R}$. We refer to η as the noise scale. The η variable controls the density of the microgeometry. We create the ground-truth using $\eta = 100$:

$$\hat{f}(\mathbf{x} \mid \eta) = \sum_{i}^{3} \sin(x_i \cdot \eta)/\eta. \tag{14}$$

The second element from the set of gyroid functions, Gyroid [3D], is guided by $\eta \in \mathbb{R}$, and $k = 2 \cdot \pi/a \in \mathbb{R}, t \in \mathbb{R}$. The a represents the cell size of the gyroid structure, t represents the wall thickness. We create the ground-truth using the setting of $\eta = 100.0$, $a = 7.0$, $t = 1.2$ as follows

$$\hat{f}(\mathbf{x} \mid \eta, k, t) = g(\mathbf{x} \cdot \eta \mid k, t)/\eta, \tag{15}$$

$$g(\mathbf{x} \mid k, t) = \sin(kx_0)\cos(kx_1) + \tag{16}$$

$$\sin(kx_1)\cos(kx_2) + \sin(kx_2)\cos(kx_0) + t. \tag{17}$$

Finally, the last exemplar, the Gyroid [5D], closely resembles the Gyroid [3D] problem. The only difference is the division of the $\mathbf{a} \in \mathbb{R}^3$ parameter into three variables $k_i = 2 \cdot \pi/a_i$, each controlling the cell size of the gyroid in the x, y, and z directions respectively. We create

the ground-truth using the setting $\eta = 100.0$, $\mathbf{a} = (7.0, 10.0, 15.0)$, $t = 1.2$. The formulation ins this case is:

$$\hat{f}(\mathbf{x} \mid \eta, \mathbf{k}, t) = r(\mathbf{x} \cdot \eta \mid \mathbf{k}, t)/\eta, \tag{18}$$

$$r(\mathbf{x} \mid \mathbf{k}, t) = \sin(k_1 x_0)\cos(k_2 x_1) + \tag{19}$$

$$\sin(k_2 x_1)\cos(k_3 x_2) + \sin(k_3 x_2)\cos(k_1 x_0) + t, \tag{20}$$

3.1.2. Spherical Microstructure

Spherical microstructure consists of an implicit function for randomly placed spheres. We generate a granular material microgeometry using a space-filling grid-based random variable of points. A grid cell index, computed from its floored location, is used to generate a seed t for the PRNG. A multilinear hash function defining the seed looks as follows:

$$t_l(\mathbf{x}) = M \sum_{i=0}^{1} \sum_{j=0}^{1} \sum_{k=0}^{1} a_{ijk} q_l(x_0)^i q_l(x_1)^j q_l(x_2)^k, \tag{21}$$

where grid indices $(q_l(x_0), q_l(x_1), q_l(x_2))$ are computed in eight directions using binary offsets and the floor operation. M and all a_{ijk} were selected such that regularity artifacts are avoided. A seed t_l is later used as an input to PRNG, generating a pseudo-random number in the interval $[0,1)$. We then add the randomly generated number to the grid position to obtain the center of a random sphere in the current grid. We repeat the described procedure a total of 8 times (once for each potential overlapping grid cell). Finally, the implicit function is calculated by finding the minimum distance to all spheres, taking into account their radius. The variables are the density-controlling $\eta \in \mathbb{R}$ and the radius of the spheres $r \in \mathbb{R}$. The procedure can be mathematically described as follows:

$$\hat{f}(\boldsymbol{x} \mid \eta, r) = \left(\min_{l=1}^{8} ||c_l(\eta \cdot \mathbf{x}, t_l(\eta \cdot \mathbf{x})) - \mathbf{x}||_2 - r \right)/\eta, \tag{22}$$

$$c_l(\eta \cdot \mathbf{x}, t_l(\eta \cdot \mathbf{x})) = \mathbf{q}_l(\eta \cdot \mathbf{x}) + \mathrm{PRNG}(t_l(\eta \cdot \mathbf{x})), \tag{23}$$

$$\mathbf{q}_l(\eta \cdot \mathbf{x}) = \lfloor \eta \cdot \mathbf{x} \rfloor + \mathcal{U}_l, \tag{24}$$

where $\mathbf{x}, t_i(\mathbf{x}), c(\mathbf{x}, t_i(\mathbf{x})), \mathcal{U}_l$ are a point of evaluation, a random seed, the center of a random sphere, and the l-th element of the unit cube vertices $\mathcal{U} = \{(v_1, v_2, v_3) \mid v_i \in \{0,1\}\}$, respectively. The PRNG is not differentiable, making $\hat{f}$ a non-differentiable function. We use the following setting to create the ground-truth $\eta = 30.0$, $r = 0.08$.

3.1.3. Fibrous Microstructure

Fibrous microstructures can be obtained by a union of two rotated sets of fibers parallel to the y-axis arranged in layers. To achieve this, we need a rotation transformation matrix T_φ along the axis y by an angle $\varphi \in [0, 2\pi]$, which is a parameter. Thanks to the symmetry of the problem, we can constrain the $\varphi \in [0, \pi]$. We also need a function $u : \mathbb{R}^3 \to \mathbb{R}$ defining the fibers. The last parameter is the $\eta \in \mathbb{R}$ controlling the density of the fibers. The microstructure Fibers [2D] is then defined by:

$$\hat{f}(\mathbf{x}|\eta, \phi) = h(\mathbf{x} \cdot \eta|\phi)/\eta, \tag{25}$$

$$h(\mathbf{x}|\phi) = \min\Big(u(\mathbf{x}) + u((x_1, x_1, x_1)), u(T_\phi(\mathbf{x})) + u(T_\phi(x_1, x_1, x_1))\Big), \tag{26}$$

$$u(x) = (\sin(x_0)\cos(x_1) + \sin(x_1)\cos(x_2) + \sin(x_2)\cos(x_0))^2. \tag{27}$$

To create the ground truth, we set $\eta = 100, \varphi = \pi/4$.

3.1.4. Porous Microstructure

Lastly, we present the function defining a porous microstructure named Porous [28D]. The first parameter, $\eta \in \mathbb{R}$, controls the density of the material. The implicit function is composed of the summation of several sine and cosine waves with different amplitudes and frequencies, resulting in a complex material structure. Three-parameter transformation matrices determine the frequencies, denoted as $\mathbf{T} \in \mathbb{R}^{3\ times 3 \times 3}$. The final distance is then computed by

$$\hat{f}(\mathbf{x}|\eta, \mathbf{T}) = \min\Big(\epsilon, v(\eta \mathbf{T}_1 \mathbf{x}) + w(\eta \mathbf{T}_2 \mathbf{x}) \cdot v(\eta \mathbf{T}_3 \mathbf{x})\Big)/\eta, \tag{28}$$

$$v(\mathbf{x}) = \Big(\sin(x_0)\cos(x_1) + \sin(x_1)\cos(x_2) + \sin(x_2)\cos(x_0)\Big)^2, \tag{29}$$

$$w(\mathbf{x}) = \Big(\sin(x_0)\sin(x_1) + \sin(x_1)\sin(x_2) + \sin(x_2)\sin(x_0)\Big)^2, \tag{30}$$

where ϵ is a small positive number. The ground truth is created with a fixed $\eta = 30$ and the $\mathbf{T} \in [-4, 7]^{3\times3\times3}$.

3.2. Parameter Fitting

This approach formulates microstructure reconstruction as a model-fitting task, where the goal is to optimize the parameters of a procedural implicit surface to match the microstructure's ground truth accurately. Given a set of spatial coordinates $\Omega = \{\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^3\}$, a ground-truth SDF $\hat{S} : \mathbb{R}^3 \to \mathbb{R}$ describing the surface at the point x, a bounded parameter space $\Phi = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$ with n parameters, and a loss function $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, the problem is defined as minimizing the discrepancy between the fitted function f and the ground truth $\hat{S}$ across the domain:

$$\min_{\boldsymbol{\phi} \in \Phi} \int_\Omega \mathcal{L}(f(\mathbf{x}, \boldsymbol{\phi}), \hat{S}(\mathbf{x}))d\mathbf{x}. \tag{31}$$

Procedural implicit surfaces allow efficient extrapolation of the microstructure beyond the optimization domain without modifying the fitted function. Additionally, our compact, parametrized microstructures are described using a relatively low number of parameters. We propose a loss function to evaluate discrepancies in the frequency domain using the discrete Fourier transform (DFT), which considers the periodic behavior common in many microstructures. The loss function compares the amplitude and phase spectra of the Fourier coefficients, using the logarithm of the mean squared error (MSE) as an error metric. This metric is closely related to the spectral density function commonly used in microstructure analysis [BZL*18].

3.3. Analysis by Synthesis

The analysis-by-synthesis approach optimizes the SDF by iteratively rendering it using a non-differentiable pipeline and comparing the results in image space. The process is illustrated in Fig. 9. Given a camera intrinsics $\mathbf{K} \in \mathbb{R}^{3\times3}$ taken from a set of intrinsic matrices $\mathcal{K}$, the camera extrinsics $\mathbf{E} = [\mathbf{R} \mid \mathbf{t}] \in \mathbb{R}^{3\times4}$ taken from a set of extrinsic matrices $\mathcal{E}$, a labeling function $\hat{S} : \mathbb{R}^3 \to \mathbb{R}$, a bounded parameter space $\Phi = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$ with n parameters, a signed distance fitting function $f : \mathbb{R}^3 \times \Phi \to \mathbb{R}$, a loss function $\mathcal{L}$:

$\mathbb{R}^{W\times H\times 3}\times\mathbb{R}^{W\times H\times 3}\to\mathbb{R}$, with W,H being width and the height respectively, a set of functions $\mathcal{A}$, and finally the rendering function $\mathcal{R}:\ \mathcal{A}\times\mathcal{K}\times\mathcal{E}\to\mathbb{R}^{W\times H\times 3}$, we can express the optimization problem as follows:

$$\min_{\boldsymbol{\phi}\in\Phi}\mathcal{L}\left(\mathcal{R}\left(f\left(\cdot,\boldsymbol{\phi}\right),\mathbf{K},\mathbf{E}\right),\mathcal{R}(\hat{S}\left(\cdot\right),\mathbf{K},\mathbf{E})\right). \tag{32}$$

$\mathcal{R}$ renders the microstructure bounded by a unit sphere from the given view with the camera parameters using the sphere-tracing algorithm [Har96] in the CUDA-based framework OptiX [PBD*10]. For the loss function, we used the two-dimensional DFT to transform the image signal into the frequency domain. By focusing on the amplitudes of the Fourier coefficients, we capture the key periodic features of the image, which are crucial for the effective global optimization of microstructures. While a multi-view optimization is possible, we opted for a single-view approach to minimize computational overhead.

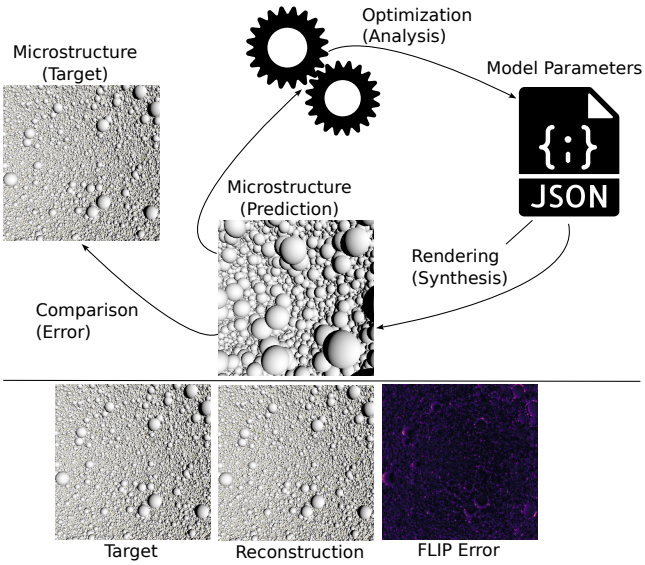

Figure 9: *Analysis by synthesis schematic. Given a single target image (such as from SEM or TEM), we perform iterative optimization. At each step, we synthesize a microstructure using an implicit function and render a corresponding synthetic image using our rendering algorithm. We then compare this synthetic image to the target image and adjust the parameters of the synthetic microstructure to reduce the difference. This process continues until the synthetic image closely resembles the target.*

3.4. Optimization Algorithms

Selecting a suitable optimization algorithm is critical for solving unconstrained global optimization problems. The loss function used in parameter fitting is differentiable, allowing us to use gradient-based optimization algorithms, as the operations involved are differentiable, including the discrete Fourier transform (DFT). In this paper, we consider two first-order methods that utilize gradient information: Basin-Hopping (BH) [Wal03] and Simplicial Homology Global Optimization (SHGO) [ESF18] algorithms. The BH algorithm is particularly effective for problems with funnel-like loss functions. Its key hyperparameters include the step size s and the temperature T, which define the bounds of the uniformly distributed random perturbations and the acceptance probability for the candidate function, respectively. SHGO global optimization algorithm approximates locally convex subdomains using homology groups. The main hyperparameters for SHGO include the number of iterations, l, and the number of samples, m, for building the complex. The Fourier loss functions are differentiable, but certain microstructures defined by procedural implicit surfaces and the rendering function $\mathcal{R}$ are non-differentiable, leading to non-differentiability of the final composite function. We consider two gradient-free optimization algorithms: the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) algorithm [HAB19] and the modified Powell method [PVTF88, Pow64]. The CMA-ES algorithm is highly effective, particularly for problems that are rugged, noisy, and non-differentiable. Its key hyperparameters are the population size $\mathcal{P}$ and the initial σ_0 of the covariance matrix, typically chosen depending on the complexity and dimensionality of the problem. The modified Powell method [PVTF88, Pow64] is a local optimizer without hyperparameters that can be effective when a reasonably close initialization is provided.

3.5. Neural networks

In this section, we describe how we use neural network to reconstruct the synthetic microstructures from incomplete exemplars of surface points and volumes.

Problem Formulation Given a set of point coordinates $\Omega=\{\mathbf{x}\mid\mathbf{x}\in\mathbb{R}^3\}$, a signed distance labeling function $\hat{S}:\ \mathbb{R}^3\ \to\ \mathbb{R}$ that provides the ground truth signed distance from the surface at a point, a parameter space Θ with h parameters, a neural network $f:\mathbb{R}^3\times\Theta\to\mathbb{R}$, and finally a loss function $\mathcal{L}:\mathbb{R}\times\mathbb{R}\to\mathbb{R}$, the problem can be formulated as follows:

$$\min_{\boldsymbol{\theta}\in\Theta}\int_{\Omega}\mathcal{L}(f(\mathbf{x},\boldsymbol{\theta}),\hat{S}(\mathbf{x}))d\mathbf{x}. \tag{33}$$

In contrast to the parameter fitting and analysis by synthesis approaches, there is no need for a manual selection of a suitable fitting function for the current problem. One drawback of the neural network is its inability to extrapolate outside of the training domain, which could be potentially solved in the future by tiling the fitted volume. In this section, we assume the use of a truncated SDF within a unit sphere, meaning that the SDF is only relevant inside the unit sphere and is truncated beyond it.

Loss Function and Optimization We adopt the loss function for optimizing 3D shapes from the SIREN paper [SMB*20], which evaluates ground truth distances and gradients only at zero-level set points, while enforcing non-zero distances elsewhere. The loss also includes the Eikonal constraint to ensure the gradient magnitude is 1, making the function behave like a true distance function. The loss is formulated as follow:

$$\mathcal{L}_{\text{sur}}=\lambda_1\int_{\Omega}|\|\nabla_{\mathbf{x}}f(\mathbf{x})\|-1|\,d\mathbf{x}+\lambda_2\int_{\Omega_0}|f(\mathbf{x})|d\mathbf{x}+$$
$$\lambda_3\int_{\Omega_0}\left(1-\nabla_{\mathbf{x}}f(\mathbf{x})^T\nabla_{\mathbf{x}}\hat{S}(\mathbf{x})\right)d\mathbf{x}+\lambda_4\int_{\Omega\setminus\Omega_0}\psi(f(\mathbf{x}))d\mathbf{x}, \tag{34}$$

where Ω_0 denotes the surface points, $\nabla_{\mathbf{x}} f(\mathbf{x})$ represents the gradient of f with respect to $\mathbf{x}$ in $\mathbf{x}$, and $\nabla_{\mathbf{x}} \hat{S}(\mathbf{x})$ here is the normal at a given surface point. The function $\psi(\mathbf{x}) = \exp(-100 \cdot |f(\mathbf{x})|)$ encourages off-surface points to maintain a non-zero distance. The first term enforces the Eikonal constraint, constraining the gradient norm to be 1. The second term forces zero-level set points to have a value of zero. The third term aligns the normals of the function f with the ground truth normals. We adopt the the hyperparameters from the paper as follows: $\lambda_1 = 50, \lambda_2 = 3000, \lambda_3 = 100, \lambda_4 = 100$. During training, for each point with a known ground truth signed distance and normal from $\mathbf{x} \in \Omega_0$, an additional random off-surface point $\mathbf{x} \in \Omega \setminus \Omega_0$ is sampled.

In the second loss function, we use the fact that the signed distance function is well-defined not only at the zero-level set points but also available at the entire volume. Therefore, we propose a loss that forces the network to learn the signed distances and gradients in the entire domain:

$$\mathcal{L}_{\text{vol}} = \lambda_1 \int_{\Omega} |\|\nabla_{\mathbf{x}} f(\mathbf{x})\| - 1| \, d\mathbf{x} + \lambda_2 \int_{\Omega} |f(\mathbf{x}) - \hat{S}(\mathbf{x})| d\mathbf{x} + \lambda_3 \int_{\Omega} \left(1 - \nabla_{\mathbf{x}} f(\mathbf{x})^T \nabla_{\mathbf{x}} \hat{S}(\mathbf{x})\right) d\mathbf{x}. \quad (35)$$

We use the same λ values as in the loss 34. Note that we do not normalize the signed distances $\hat{S}(\mathbf{x})$. To find the minima of the loss functions 34, 35, we use the ADAM optimizer [KB14] with the $\alpha = 10^{-4}$ and $\beta_1 = 0.9$, $\beta_2 = 0.999$ for all our experiments.

Data Generation To generate the ground truth signed distances, we use the same approach as the synthetic dataset. For the optimization of neural networks, we also need to compute the gradients with respect to the inputs. For the synthetic microstructures that are not differentiable (e.g., spherical microstructures), we approximate the gradients using the central differences scheme. We consider two scenarios: surface-level points and volume-level points. Specifically, a point $\mathbf{x}$ is considered to belong to the surface Ω_0 if its signed distance satisfies $|f(\mathbf{x})| < \epsilon$, with $\epsilon = 10^{-3}$.

Network Architectures We consider two neural network architectures. The first architecture is a standard MLP with ReLU as the activation function, where we also use the positional encoding of the input coordinates, adapted from NeRF [MST*21]. This enables the network to better represent higher frequency signals, such as sharp details of the microstructure. We initialize the network using the He initialization [HZRS15] and fix the number of hidden layers at five, experimenting with 256 and 512 features per layer. The second network architecture uses a periodic sine activation function instead of the standard ReLU [SMB*20]. We expect the periodic component, similarly to positional encoding, would help the neural network to encode the repetitive patterns of the microstructures. We use the same initialization with *omega* = 30 as in the original SIREN paper [SMB*20]. To ensure a fair comparison with the MLP using Positional Encoding, we also fixed the number of hidden layers at five and experimented with 256 and 512 features per layer.

3.6. Reconstruction Results

In Fig. 10, we show the initial structures for the reconstruction experiments. In Table 2, we report the performance of different algorithms for different microstructures. In Table 3, we report the initial parameters and the optimized parameters after the reconstruction. In Fig. 11, we show the corresponding renders. We set a maximum time limit of approximately 20 minutes for optimization, as results typically do not improve significantly beyond this point. The results show that no single optimization algorithm consistently outperforms the others across all microstructures. SHGO-PF and BH-PF perform well for small-variable problems but struggle with higher-dimensional cases (e.g., Gyroid [5D], Porous [28D]) and cannot be applied to certain discrete problems (e.g., Spheres [2D]). CMA-ES-PF demonstrates solid performance across various microstructures, including discrete ones such as spheres [2D]. As expected, parameter fitting methods outperform analysis-by-synthesis in both accuracy and efficiency, given the larger complexity of the latter. Interestingly, CMA-ES-AS resembles Porous [28D] more than the other methods, highlighting its potential for high-dimensional reconstructions.

In Table 4, we report the performance of different neural network architectures. In Fig. 12 and Fig. 13, we include the corresponding renders for the volume and surface reconstruction results, respectively. Notice that all neural network architectures struggle to recover the shapes given surface or volumetric points for the Fibers [2D], Spheres[2D], and Porous [28D] microstructures. The discrete nature of these microstructures and the non-differentiable search spaces make the optimization with neural networks particularly challenging. The results show that the SIREN architecture outperforms the MLP with Positional Encoding architecture in most microstructures, particularly in recovering thin structures. Hybrid models combining the strengths of both classical optimization methods and neural networks present an interesting venue for future work.

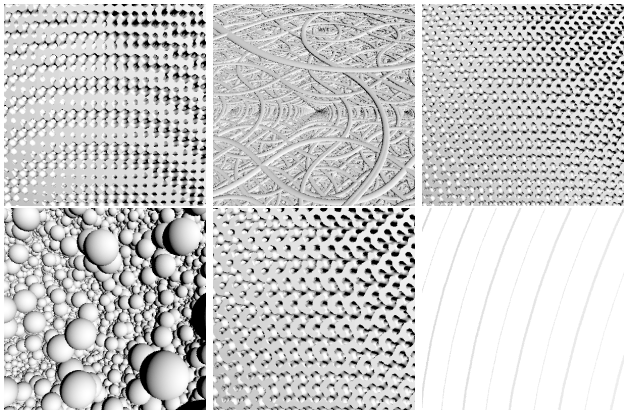

Figure 10: *Initial structures (top row) Gyroid [1D] 14, Fibers [2D] 25, Gyroid [3D] 15 respectively. Rest of the initial structures (bottom row) are Spheres [2D] 22, Gyroid [5D] 18, Porous [28D] 28 respectively.*

Microstructure	Metrics	SHGO-PF	BH-PF	CMA-ES-PF	CMA-ES-AS	Powell-AS
Gyroid [1D]	Val. Error↓	**0.00**	**0.00**	**0.00**	10^{-4}	10^{-6}
	LPIPS↓	**0.00**	**0.00**	**0.00**	0.02	**0.00**
	∄LIP ↓	3×10^{-7}	$\mathbf{1\times10^{-7}}$	1×10^{-3}	2×10^{-3}	2×10^{-3}
	Time↓	**2.85s**	16.67s	**2.23s**	20m	7m
Fibers [2D]	Val. Error↓	0.34	$\mathbf{1\times10^{-3}}$	0.65	1.00	1.00
	LPIPS↓	0.41	**0.13**	0.54	0.56	0.56
	∄LIP ↓	0.3	**0.14**	0.42	0.40	0.43
	Time↓	13m	20m	6m	20m	**5m**
Gyroid [3D]	Val. Error↓	$\mathbf{1\times10^{-3}}$	6×10^{-3}	1.02	0.15	2×10^{-3}
	LPIPS↓	**0.01**	0.04	0.65	0.35	0.02
	∄LIP ↓	**0.01**	0.05	0.61	0.42	0.03
	Time↓	**22.7s**	20m	7m	14m	20m
Spheres [2D]	Val. Error↓	~~N/A~~	~~N/A~~	$\mathbf{1\times10^{-5}}$	1.01	1.01
	LPIPS↓	~~N/A~~	~~N/A~~	**0.00**	0.53	0.53
	∄LIP ↓	~~N/A~~	~~N/A~~	$\mathbf{3\times10^{-7}}$	0.55	0.55
	time↓	~~N/A~~	~~N/A~~	**4m**	20m	5m
Gyroid [5D]	Val. Error↓	0.66	0.65	$\mathbf{3\times10^{-4}}$	0.03	0.44
	LPIPS↓	0.42	0.57	**0.00**	0.21	0.54
	∄LIP ↓	0.42	0.62	$\mathbf{6\times10^{-3}}$	0.28	0.60
	Time↓	**2m**	15m	11m	20m	15m
Porous [28D]	Val. Error↓	~~N/A~~	**1.00**	**1.00**	**1.00**	**1.00**
	LPIPS↓	~~N/A~~	0.64	0.64	0.64	**0.61**
	∄LIP ↓	~~N/A~~	**0.51**	0.52	0.52	0.59
	Time↓	~~N/A~~	**4m**	8m	20m	20m

Table 2: *Reconstruction performance for different optimization algorithms (columns) and synthetic procedural microstructures (rows). We report for each microstructure the **a)** Validaton Error, **b)** LPIPS Perceptual loss, **c)** ∄LIP error, **d)** Time of the optimization. We consider a numerical 0 to be a number below 1×10^{-7}. The ~~N/A~~ implies that the optimization method could not be applied for the corresponding microstructure. The best results are highlighted in orange, while the second-best results are marked in yellow.*

Problem	Ground Truth	Init.	SHGO-PF	BH-PF	CMA-ES-PF	CMA-ES-AS	Powell-AS
Gyroid [1D]	η: 100.0	300.0	**100.000**	**100.000**	**100.000**	100.001	**100.000**
Fibers [2D]	η: 100.0	200.0	**100.000**	**100.000**	100.100	175.159	133.202
	θ : π/4	π	0.142	**0.785**	2.825	3.126	0.831
Gyroid [3D]	η: 100.0	300.0	**116.633**	213.469	32.000	186.593	86.780
	a : 7.0	6.0	8.164	14.942	11.010	12.958	**6.075**
	t : 1.2	0.0	**1.200**	**1.200**	1.196	1.173	1.198
Spheres [2D]	η: 30.0	150.0	~~N/A~~	~~N/A~~	**29.999**	291.739	300.964
	r : 0.08	0.2	~~N/A~~	~~N/A~~	**0.079**	0.093	0.086
Gyroid [5D]	η: 100.0	200.0	114.983	142.570	**97.668**	74.162	89.104
	a : 7.0	6.0	8.051	9.979	**6.836**	5.180	6.319
	t : 10.0	6.0	11.534	14.162	**9.766**	7.376	9.228
	t : 15.0	6.0	**14.961**	14.778	14.650	11.062	13.125
	t : 1.2	6.0	1.180	1.199	**1.200**	1.165	1.151
Porous [28D]	η: 30.0	50.0	~~N/A~~	38.586	42.796	**28.268**	43.272

Table 3: *Optimal function parameters $\boldsymbol{\phi}$ Table. Optimal parameters found by the specified global optimizers. These are the parameters used to compute the error metrics in Table 2. The parameters were rounded to 3 decimal points. We also specify the ground truth and the initialization of the variables.*

3.7. Differentiable SDF Rendering

We verify that the current state of the art for differentiable physically-based SDF rendering [VSJ22] does not handle typical features for multiscale material reconstruction, such as multi-object reconstruction and porous structures. In Figure 14, we show an example of failures in the reconstruction of a small set of objects (two spheres in contact and three cubes) and a porous sphere. In addition, it is not always possible to guarantee that the microstructure mathematical formulation is always differentiable, making differentiable optimization not possible in some scenarios. Further investigation to extend the algorithm to support multi-object optimization is an interesting opportunity for future work.

Microstructure		SIREN-SUR-256	SIREN-SUR-512	SIREN-VOL-256	SIREN-VOL-512	MLP-PE-SUR-256	MLP-PE-SUR-512	MLP-PE-VOL-256	MLP-PE-VOL-512
Gyroid [1D]	Val. Error↓	0.03	0.03	0.02	**0.01**	0.04	0.15	0.13	0.36
	LPIPS↓	0.42	0.31	0.21	**0.19**	0.47	0.65	0.37	0.49
	∄LIP ↓	0.28	0.22	0.18	**0.16**	0.32	0.46	0.19	0.25
	max $\|\|\nabla f(x)\|\|$ ↕ 1	2.18	1.95	2.07	**1.91**	2.51	4.81	7.04	11.94
	Δ-loss↑	130.31	149.31	289.02	387.23	17650.59	16391.40	23623.22	22462.19
Fibers [2D]	Val. Error↓	1.00	1.00	1.00	1.00	1.07	0.60	**0.39**	0.43
	LPIPS↓	0.72	**0.69**	**0.69**	**0.69**	**0.69**	**0.69**	0.71	0.73
	∄LIP ↓	0.95	0.51	0.49	0.54	0.51	0.51	0.38	**0.37**
	max $\|\|\nabla f(x)\|\|$ ↕ 1	3.58	4.73	5.43	3.88	**2.08**	2.79	2.75	4.31
	Δ-loss↑	66.99	88.40	208.28	274.95	20743.34	17657.37	20647.70	23504.41
Gyroid [3D]	Val. Error↓	0.12	0.10	0.09	**0.05**	0.06	0.13	0.08	0.15
	LPIPS↓	0.41	**0.34**	0.61	0.59	0.47	0.52	0.65	0.72
	∄LIP ↓	0.46	**0.40**	0.58	0.56	0.46	0.46	0.57	0.58
	max $\|\|\nabla f(x)\|\|$ ↕ 1	3.41	2.86	2.83	3.42	**1.99**	2.77	2.90	3.69
	Δ-loss↑	121.56	178.95	311.83	412.26	21315.19	19604.08	24535.98	20033.85
Spheres [2D]	Val. Error↓	0.70	0.94	0.46	**0.36**	1.02	1.00	1.00	1.01
	LPIPS↓	0.66	0.64	0.61	**0.59**	0.66	0.66	0.64	0.64
	∄LIP ↓	0.57	0.58	**0.54**	**0.54**	**0.54**	**0.54**	**0.54**	**0.54**
	max $\|\|\nabla f(x)\|\|$ ↕ 1	4.61	4.98	**2.67**	4.55	4.7	10.59	15.45	19.13
	ϵ-loss↑	133.71	238.88	345.84	544.73	23633.08	21651.34	24528.74	25124.24
Gyroid [5D]	Val. Error↓	0.11	0.08	**0.03**	0.04	1.02	1.04	0.46	0.30
	LPIPS↓	0.33	0.35	**0.31**	0.33	0.67	0.67	0.66	0.69
	∄LIP ↓	0.38	0.41	**0.36**	0.38	0.61	0.61	0.60	0.57
	max $\|\|\nabla f(x)\|\|$ ↕ 1	2.66	2.37	2.81	**1.61**	2.89	2.89	4.41	4.40
	Δ-loss↑	138.95	167.09	244.68	383.13	21584.95	17986.02	20003.56	22752.97
Porous [28D]	Val. Error↓	0.91	1.00	0.79	**0.70**	0.92	0.94	1.00	1.00
	LPIPS↓	0.63	0.63	**0.6**	**0.6**	0.77	0.71	0.79	0.8
	∄LIP ↓	0.47	0.47	0.55	0.51	**0.46**	0.56	0.59	0.47
	max $\|\|\nabla f(x)\|\|$ ↕ 1	3.77	3.92	4.23	4.18	**2.75**	3.19	3.57	3.92
	Δ-loss↑	49.25	86.32	271.66	393.56	15319.12	17610.17	22234.11	22713.46

Table 4: *Trained Networks. Table of comparison of the network architectures and the training domains used on our sythetic dataset 3.1. We show for each problem the following:* ***a)*** *Validaton Error,* ***b)*** *LPIPS Perceptual loss,* ***c)*** *Image ∄LIP error,* ***d)*** *The maximum norm of the gradient inside of the learning domain,* ***e)*** *The difference between the loss at the beginning and at the end of the optimization process denoted as Δ-loss. The best results are highlighted in orange, while the second-best results are marked in yellow.*

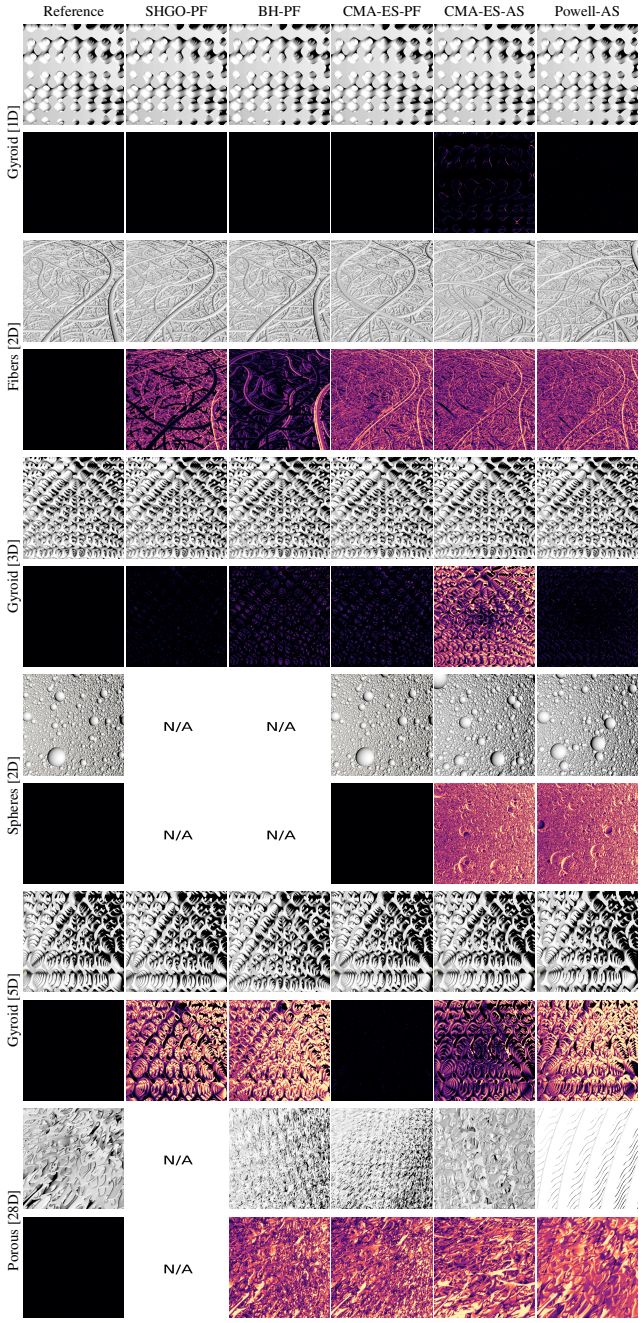

Figure 11: *Reconstruction results for different optimization algorithms (columns) and synthetic procedural microstructures (rows). We show a comparison of rendered ground truth exemplars and the rendered approximations (top row per microstructure) with the measured ϡLIP (bottom row per microstructure).*

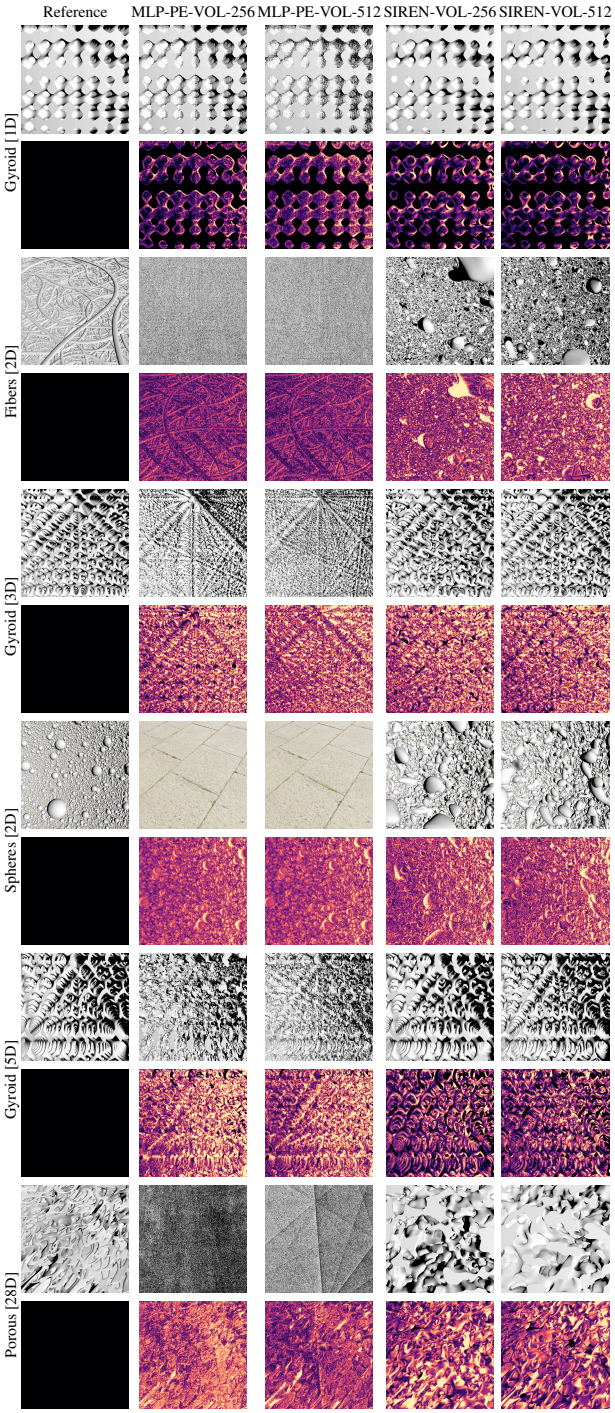

Figure 12: *Volume reconstruction results for different neural network architectures (columns) and synthetic procedural microstructures (rows). We show a comparison of rendered ground truth exemplars and the rendered approximations (top row per microstructure) with the measured ϡLIP (bottom row per microstructure).*

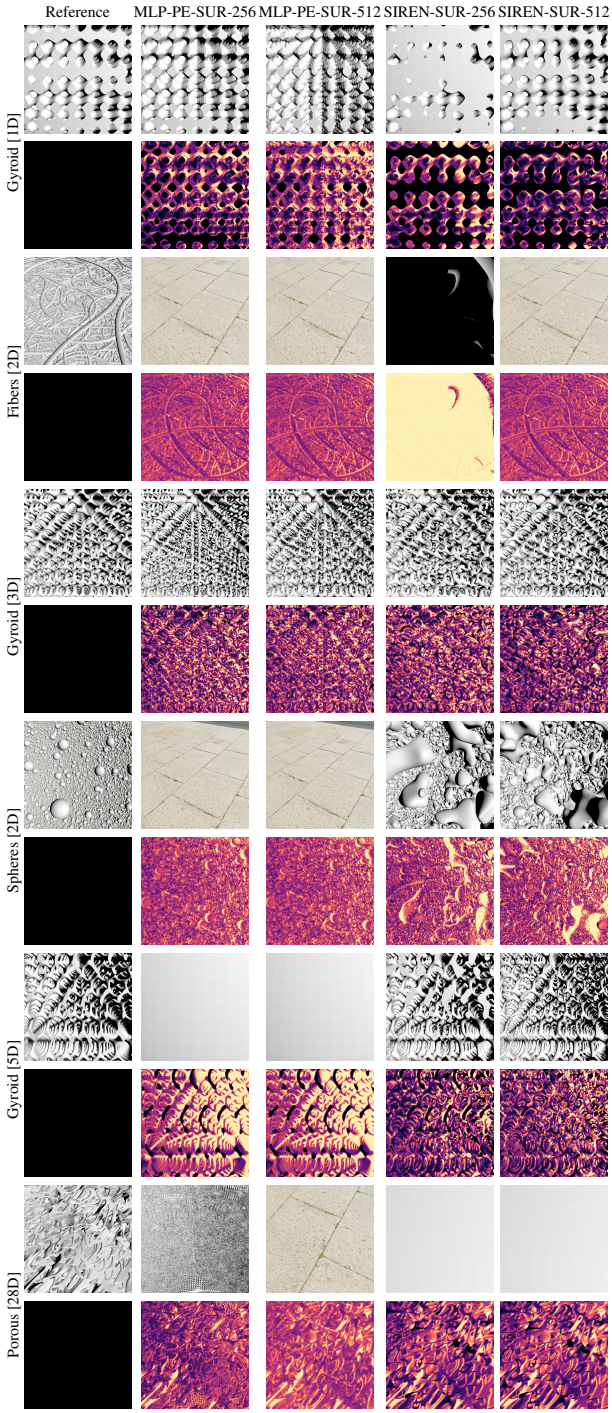

Figure 13: *Surface reconstruction results for different neural network architectures (columns) and synthetic procedural microstructures (rows). We show a comparison of rendered ground truth exemplars and the rendered approximations (top row per microstructure) with the measured ∇LIP (bottom row per microstructure).*

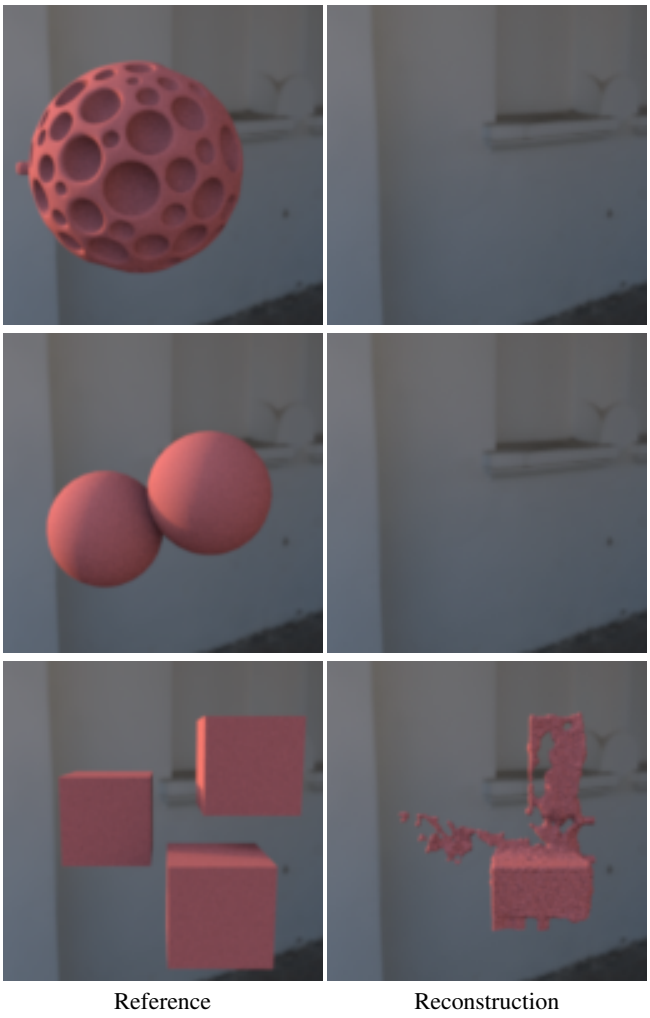

Figure 14: *Results using a differentiable physically-based SDF rendering algorithm [VSJ22] for microstructure reconstruction including typical features such as multiple objects and porous structures. The algorithm fails after 512 iterations to reconstruct all cases, including empty solution for the first two rows and visible artifacts for the last row. The initialization is the unit sphere.*

References

[BZL*18] BOSTANABAD R., ZHANG Y., LI X., KEARNEY T., BRINSON L. C., APLEY D. W., LIU W. K., CHEN W.: Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques. *Progress in Materials Science 95* (2018), 1–41. 6

[ESF18] ENDRES S. C., SANDROCK C., FOCKE W. W.: A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization 72* (2018), 181–217. 7

[FW07] FRISVAD J. R., WYVILL G.: Fast high-quality noise. In *Proceedings of GRAPHITE 2007* (2007), pp. 243–248. doi:10.1145/1321261.1321305. 3

[Ger04] GERALD C. F.: *Applied numerical analysis*. 2004. 3

[HAB19] HANSEN N., AKIMOTO Y., BAUDIS P.: CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, Feb. 2019. 7

[Har96] HART J. C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12*, 10 (1996), 527–545. doi:10.1007/s003710050084. 3, 7

[HC18] HAN L., CHE S.: An overview of materials with triply periodic minimal surfaces and related geometry: From biological structures to self-assembled systems. *Advanced Materials 30*, 17 (2018), 1705708. 2

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034. 8

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 8

[MST*21] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM 65*, 1 (2021), 99–106. 5, 8

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., ET AL.: Optix: a general purpose ray tracing engine. *Acm transactions on graphics (tog) 29*, 4 (2010), 1–13. 7

[PH89] PERLIN K., HOFFERT E. M.: Hypertexture. *Computer Graphics (SIGGRAPH '89) 23*, 3 (July 1989), 253–262. doi:10.1145/74333.74359. 2

[Pow64] POWELL M. J.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal 7*, 2 (1964), 155–162. 7

[PVTF88] PRESS W. H., VETTERLING W. T., TEUKOLSKY S. A., FLANNERY B. P.: *Numerical recipes*. 1988. 7

[SMB*20] SITZMANN V., MARTEL J. N., BERGMAN A. W., LINDELL D. B., WETZSTEIN G.: Implicit neural representations with periodic activation functions. In *Proc. NeurIPS* (2020). 5, 7, 8

[TEZ*19] TRICARD T., EFREMOV S., ZANNI C., NEYRET F., MARTÍNEZ J., LEFEBVRE S.: Procedural phasor noise. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–13. 1

[VSJ22] VICINI D., SPEIERER S., JAKOB W.: Differentiable signed distance function rendering. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–18. 5, 9, 12

[Wal03] WALES D.: Energy landscapes (cambridge molecular science), 2003. 7