Neural Implicit Surface Evolution
– Supplementary Material –

Tiago Novello
IMPA

Vinicius da Silva
PUC-Rio

Guilherme Schardong
U Coimbra

Luiz Schirmer
Unisinos

Helio Lopes
PUC-Rio

Luiz Velho
IMPA

1. Minimal surfaces

The evolution of a surface S governed by the *mean curvature equation* (MCE) leads to a family of surfaces that reduce their area over time. Let $f:\mathbb{R}^3\times\mathbb{R}\to\mathbb{R}$ be a solution of MCE and S_t be its corresponding surface evolution.

$$\begin{cases} \dfrac{\partial f}{\partial t} - \alpha \left\|\nabla f\right\| \kappa = 0 & \text{in } \mathbb{R}^3 \times (a,b), \\ f = g & \text{on } \mathbb{R}^3 \times \{t=0\}. \end{cases} \tag{1}$$

The area of S_t can be measured using $\text{Area}(S_t) = \int_{S_t} dS_t$, where dS_t is the area form of S_t. It can be proved that the *first variation of area* of the family S_t is given by

$$\frac{d}{dt}\text{Area}(S_t)\Big|_{t=0} = -\int_{S_0} \kappa^2 dS_0, \tag{2}$$

The proof can be found in [1, Sec. 3.5], [3, Cor. 6.2]. Thus, if the mean curvature satisfies $\kappa \neq 0$, the area of S_t initially decreases because its derivatives are negative at $t=0$.

A surface S_{t_0} is *critical* if $\frac{d}{dt}\text{Area}(S_t)\Big|_{t=t_0} = 0$, that is, if κ is constant equal to zero. This surface is called *minimal*. Examples of minimal surfaces include the plane, catenoids, helicoids, Enneper surface, Costa's minimal surfaces, etc.

We can fix a region of the initial surface S in the MCE. If S has a boundary curve, fixing it during the evolution leads to a surface of minimal area. This problem is related to the physical shapes of soap films at equilibrium under the surface tension [5].

2. Network initialization

Proposition 2.1. *Let $g_\phi : \mathbb{R}^3 \to \mathbb{R}$ and $f_\theta : \mathbb{R}^3\times\mathbb{R} \to \mathbb{R}$ be networks with depth d. If f_θ is wider than g_ϕ, we can define θ in terms of ϕ such that $f_\theta(p,t) = g_\phi(p)$ for all (p,t).*

Proof. Recall that $g_\phi(p) = B_{d+1} \circ g_d \circ \cdots \circ g_1(p) + b_{d+1}$, where $g_i(p_i) = \sin(B_i p_i + b_i)$ is the i-layer, B_i is a matrix in $\mathbb{R}^{N_{i+1}\times N_i}$, and $b_i \in \mathbb{R}^{N_{i+1}}$ is the i-bias. Analogously, $f_\theta(p,t) = A_{d+1} \circ f_d \circ \cdots \circ f_1(p,t) + a_{d+1}$, with its i layer $f_i : \mathbb{R}^{M_i} \to \mathbb{R}^{M_{i+1}}$ given by $f_i(p_i) = \sin(A_i p_i + a_i)$.

By hypothesis, f_θ, and g_ϕ have the same depth d, and the width of each layer of g_ϕ is less than or equal to the width of the respective layer of f_θ, i.e., $N_i \leq M_i$. Thus, we define the hidden layers of f_θ using $A_i = \begin{pmatrix} B_i & 0 \\ 0 & 0 \end{pmatrix}$, $a_i = \begin{pmatrix} b_i \\ 0 \end{pmatrix}$. Evaluating $(p,c) \in \mathbb{R}^{N_i} \times \mathbb{R}^{M_i - N_i}$ in f_i results in

$$f_i(p,c) = \sin\left(\begin{pmatrix} B_i & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} p \\ c \end{pmatrix} + \begin{pmatrix} b_i \\ 0 \end{pmatrix}\right) = \begin{pmatrix} g_i(p) \\ 0 \end{pmatrix}.$$

Thus, defining $A_1 = \begin{pmatrix} B_1 & 0 \\ F_p & F_t \end{pmatrix}$, $a_1 = \begin{pmatrix} b_1 \\ 0 \end{pmatrix}$, we obtain the desired result because

$$f_1(p,c) = \sin\left(\begin{pmatrix} B_1 & 0 \\ F_p & F_t \end{pmatrix}\begin{pmatrix} p \\ t \end{pmatrix} + \begin{pmatrix} b_i \\ 0 \end{pmatrix}\right) = \begin{pmatrix} g_1(p) \\ F_p c + F_t t \end{pmatrix}.$$

In other words, the neurons $g_i(p)$ of the network g_ϕ remain intact along the layers. $\square$

The blocks F_p and F_t project the entry points (p,t) to a dictionary of sine waves, which are not considered in the following layers because they are fed to zero blocks. However, new hidden weights can activate such features as the training advances. In Sec 4.4, we present experiments varying the width of f_θ to explore such initialization in the problem of solving the MCE.

To reproduce Prop 2.1 with f_θ deeper than g_ϕ, we must be able to add a hidden layer $f(p_i) = \sin(Ap + a)$ to f_θ which do not exist in g_ϕ. Thus, it would be desirable to initialize f as an identity layer. Following the above approach, we could define $A = I$ and $a = 0$ obtaining $f(p) = \sin(p)$, however, in general, $\sin(p) \neq p$. This can be fixed using that $\sin(p) \approx p$ when $\|p\|$ is close to zero. Therefore, we define $A = \lambda I$, with λ being a small number, and multiply the resulting output of f by $\frac{1}{\lambda}$ to keep it close to p.

3. Extracting a network at a given time instant

Here, for a given time instant t, we extract the network $g_\phi = f(\cdot,t) : \mathbb{R}^3 \to \mathbb{R}$ from a neural network $f_\theta : \mathbb{R}^3\times\mathbb{R}\to\mathbb{R}$. Suppose $f_\theta(p,t) = A_{d+1} \circ f_d \circ \cdots \circ f_1(p,t) + a_{d+1}$, with each i-layer defined by $f_i(p_i) = \sin(A_i p_i + a_i)$.

To define g_ϕ such that $g_\phi(p) = f_\theta(p,t)$ for all (p,t), we modify the first layer $f_1(p,t) = \sin(A_1(p,t) + a_1)$ of f_θ.

The matrix A_1 has 4 column vectors $\{w_1, w_2, w_3, u\}$ in $\mathbb{R}^{M_2}$, where M_2 is the dimension of the codomain of f_1. Denoting p by (x, y, z), we obtain

$$A_1(p, t) = x \cdot w_1 + y \cdot w_2 + z \cdot w_3 + t \cdot u,$$

We use the matrix B_1 consisting of the columns w_1, w_2, w_3, and the bias $b_1 = a_1 + t \cdot u$ to set the first layer g_1 of g_ϕ. Specifically, we define g_ϕ through:

$$g_\phi(p) = A_{d+1} \circ f_d \circ \cdots \circ f_2 \circ g_1(p) + a_{d+1}.$$

Note that g_ϕ equals f_θ, except for its first layer $f_1(p, t)$, which is replaced by $g_1(p)$. We define it as

$$g_1(p) = \sin\Big(\underbrace{x \cdot w_1 + y \cdot w_2 + z \cdot w_3}_{B_1 p} + \underbrace{t \cdot u + a_1}_{b_1}\Big).$$

From the definition of g_ϕ, we have $g_\phi(p) = f_\theta(p, t)$, which implies a kind of opposite direction of Prop 2.1.

Proposition 3.1. *Let $f_\theta : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ be a neural network, and $t \in \mathbb{R}$. There is a network $g_\phi : \mathbb{R}^3 \to \mathbb{R}$ with the same hidden layers of f_θ such that $f_\theta(p, t) = g_\phi(p)$ for all $p \in \mathbb{R}^3$.*

4. Ablation studies

The ablation studies detailed below were performed using the MCE (Eq 1) under two settings: With our initialization scheme, presented in Prop 2.1, and the standard initialization [4]. The goal is to compare the training convergence of the *data constraint* $\mathcal{L}_{\text{data}}$ and the *LSE constraint* $\mathcal{L}_{\text{LSE}}$ for both initialization schemes under different circumstances.

We will visualize the graphs of $\mathcal{L}_{\text{data}}$ and $\mathcal{L}_{\text{LSE}}$ during the training of a neural network $f_\theta : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ to satisfy the MCE within a time interval (a, b). In the upcoming experiments, we will use the SDF $g : \mathbb{R}^3 \to \mathbb{R}$ of the Bunny as the initial condition, $f = g$ on $\mathbb{R}^3 \times \{0\}$. To initialize f_θ using our method, we approximate g by a network g_ϕ.

4.1. Varying time interval

We vary (a, b) in the MCE with scale $\alpha = 0.001$. We recall that the initial condition is at $0 \in (a, b)$ and on the positive (negative) part, the MCE smooths (sharpens) it. Thus, the positive part should be easier to train since no higher frequencies would arise. In contrast, training the negative part creates new higher frequencies, which could take longer to learn. We evaluate it in the following intervals:

$$\begin{aligned}(a, b) = &(0, 0.25), (0, 0.5), (0, 1),\\ &(-0.1, 0.25), (-0.1, 0.5), (-0.1, 1),\\ &(-0.25, 0.25), (-0.25, 0.5), (-0.25, 1).\end{aligned}$$

Fig 1 and 2 present the data and LSE constraint convergences for these intervals. As expected, our initialization (top image) provides a better training convergence. For $\mathcal{L}_{\text{data}}$, this is due to the fact that $f_\theta = g_\phi$ at $t = 0$, thus, $\mathcal{L}_{\text{data}}$ only have to maintain this restriction.

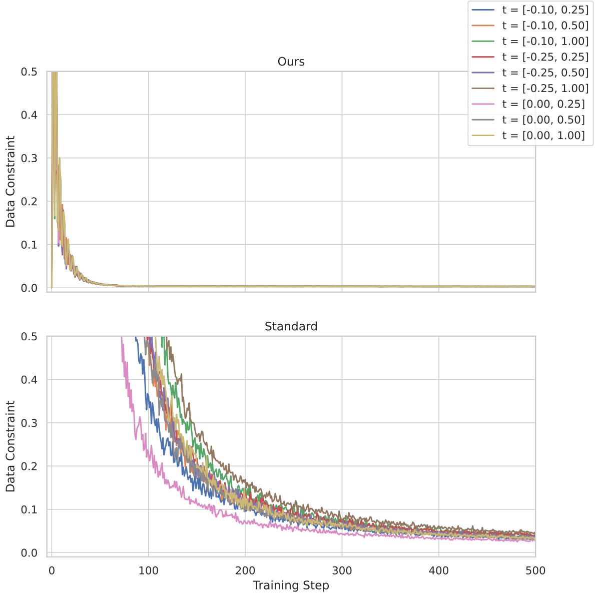

Figure 1. Data constraint values for different training intervals.

The convergence of the constraints $\mathcal{L}_{\text{data}}$ and $\mathcal{L}_{\text{LSE}}$ is faster for both initializations when using smaller intervals, as can be seen in the case of $(0, 0.25)$ (in purple). They also take longer to train on intervals with a negative part. This is likely because the solutions in such regions are sharper, requiring, thus, more frequencies for accurate representation, if a solution exists at all.

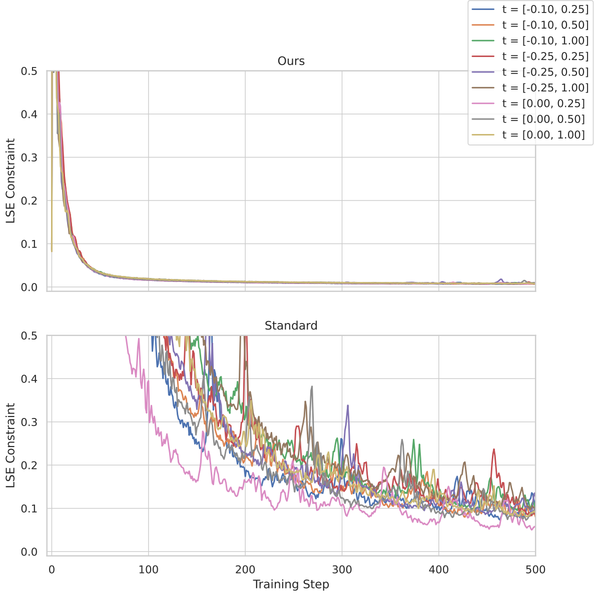

Figure 2. LSE constraint values for different training intervals.

4.2. Varying MCE scale

We use the interval $(a, b) = (-0.1, 1)$ and vary the scales $\alpha = i \times 10^{-3}$ for $i = 1, 2, 3, 4, 5, 10, 100$. In theory, increasing (a, b) while fixing α is equivalent to the previous experiment. However, in practice, the representation capacity of f_θ may not be enough to learn large variations in a short time period. This is evident in Figs 3-4, where the convergence of $\mathcal{L}_{\text{data}}$ and $\mathcal{L}_{\text{LSE}}$ is sorted by α. In general, our initialization results in a better convergence, but we observed that when using a high scale $\mathcal{L}_{\text{data}}$ diverges first, since $\mathcal{L}_{\text{LSE}}$ dominates the training. See the case $\alpha = 0.1$ (in purple).

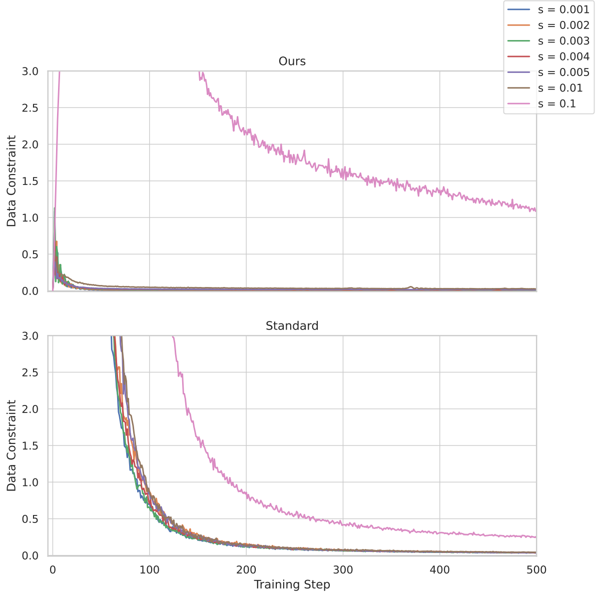

Figure 3. Data constraint values for different MCE scale values.

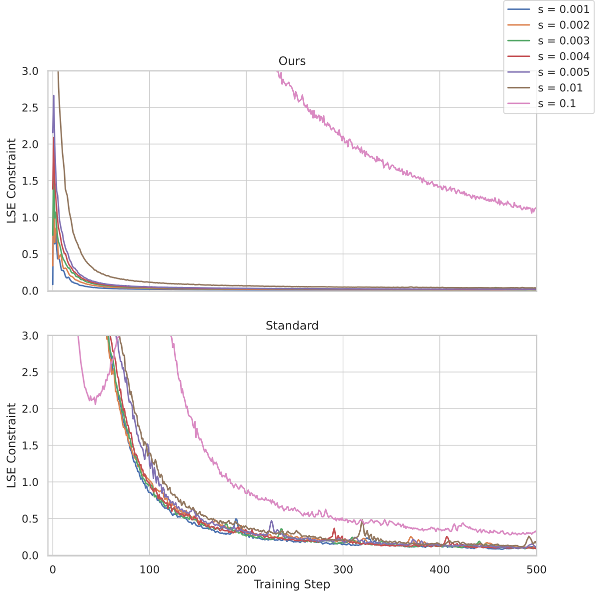

Figure 4. LSE constraint values for different MCE scale values.

4.3. Varying the point-sampling proportions

This experiment aimed to evaluate how the point-sampling of initial and intermediate conditions impacts the training convergence of $\mathcal{L}_{\text{data}}$ and $\mathcal{L}_{\text{LSE}}$. We used the default point-sampling proportions of $\{l_1, l_2, l_3\} = \{0.25, 0.25, 0.5\}$, as well as $\{0.1, 0.1, 0.8\}$ and $\{0.4, 0.4, 0.2\}$. Here, l_1, l_2, and l_3 are the numbers of space-time, on-surface, and off-surface points sampled at each training step (see Sec 4.2 of the main paper).

Figs 5-6 present the convergences of the resulting constraints during training. It can be observed that sampling fewer points at $t = 0$ results in a better convergence.

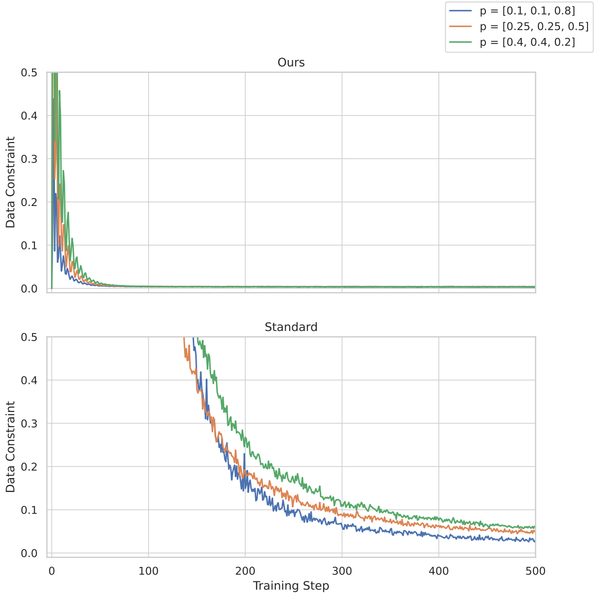

Figure 5. $\mathcal{L}_{\text{data}}$ values for different sampling proportions.

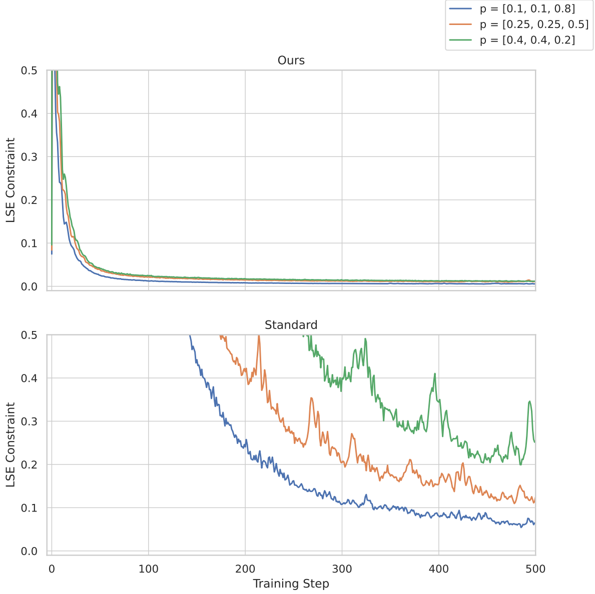

Figure 6. $\mathcal{L}_{\text{LSE}}$ values for different sampling proportions.

However, when using different sampling proportions, we obtain new constraints $\mathcal{L}_{\text{data}}$ and $\mathcal{L}_{\text{LSE}}$. Also, sampling fewer points at $t = 0$ can result in a longer convergence time for $\mathcal{L}_{\text{data}}$, as shown in the initial condition (Bunny) for each proportion in Fig 7. This is probably due to the *spectral bias* phenomenon: lower frequencies are learned first. As a result, $\mathcal{L}_{\text{LSE}}$ benefits from having a smoother initial condition and prevents fitting at $t = 0$.

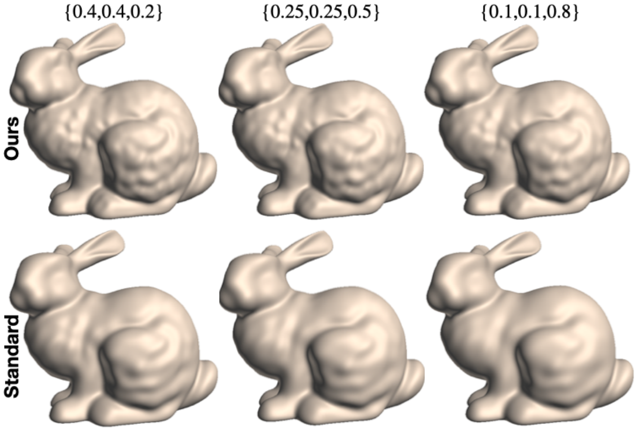

Figure 7. The zero-level sets of f_θ at $t = 0$ trained using the proportions $\{0.1, 0.1, 0.8\}$, $\{0.25, 0.25, 0.5\}$, and $\{0.4, 0.4, 0.2\}$.

4.4. Varying the network width

This experiment evaluates the impact of the network width on the training convergence using our standard initializations. We began with a width of 128 neurons and increased it by 16 neurons to a limit of 256. The remaining parameters are set to $(a, b) = (-0.25, 1)$, $\alpha = 1e-3$, $\{l_1, l_2, l_3\} = \{0.25, 0.25, 0.5\}$. As expected, increasing the width leads to better convergence; see Figs 8-9.

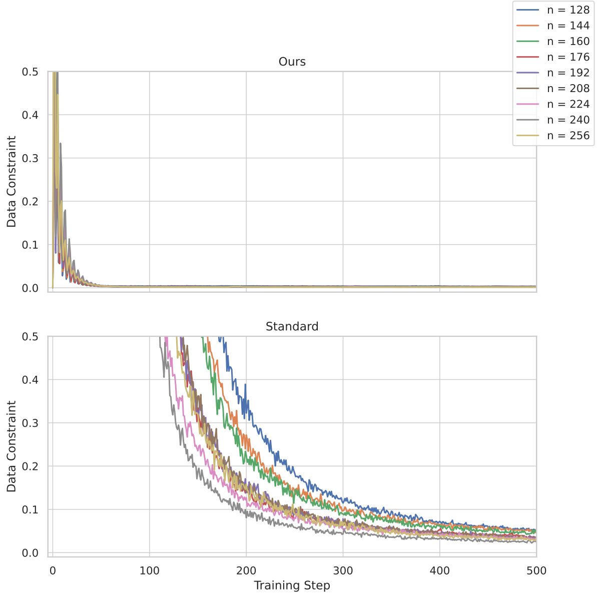

Figure 8. Data constraint values for different network widths.

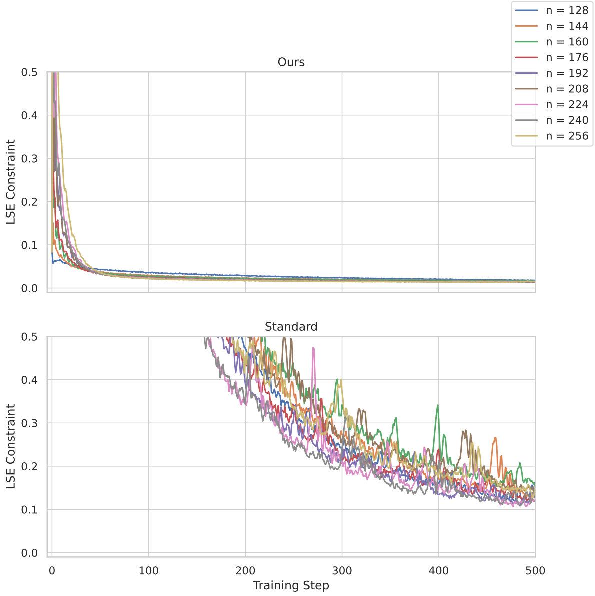

Figure 9. LSE constraint values for different network widths.

4.5. Varying the initial condition

Finally, we vary the initial condition of the MCE to evaluate the convergence of the network f_θ on different models: Bob, Max Planck, Falcon, Witch, and Neptune. During the training of each model, we fix the sampling proportions to $\{l_1, l_2, l_3\} = \{0.25, 0.25, 0.5\}$. Table 1 presents the network architectures (the width of the hidden layers), the time required for training 500 epochs, the animation interval (a, b), and the MCE scale α parameters.

Model	Network arch.	Interval	Scale	Time (s)
Bob	$[64, 64]$	$(-0.5, 1)$	$1e-2$	5.04
Max	$[128, 128, 128]$	$(-0.5, 1)$	$2e-3$	7.07
Falcon	$[160, 160, 160]$	$(-0.1, 1)$	$1e-3$	112.17
Witch	$[256, 256, 256]$	$(-0.5, 1)$	$1e-3$	143.34
Neptune	$[300, 300, 300]$	$(-0.1, 1)$	$2e-4$	162.42

Table 1. The network architectures and the time spent in their training to learn the evolution of the Bob, Max Planck, Falcon, Witch, and Neptune surfaces under the MCE.

For the sampling of on-surface points, we used different point clouds sampled from the original models. This affects the time needed to train our networks, as each epoch is defined as a complete iteration over the point-cloud. The Bob, Max Planck, Falcon, Witch, Neptune have 5344, 5002, 72466, 77553, 72668 points, respectively.

Fig 10 illustrates the zero-level sets of the resulting evolutions of Bob, Max Plank, Falcon, Witch, and Neptune models (middle). The sharpened models are on the left column. Notice that their geometric features are enhanced. Particularly, Max Planck's nose, mouth and ears are noticeably more prominent. The same occurs for the Wizard's sword and cape, and Neptune's hands and spear tip.

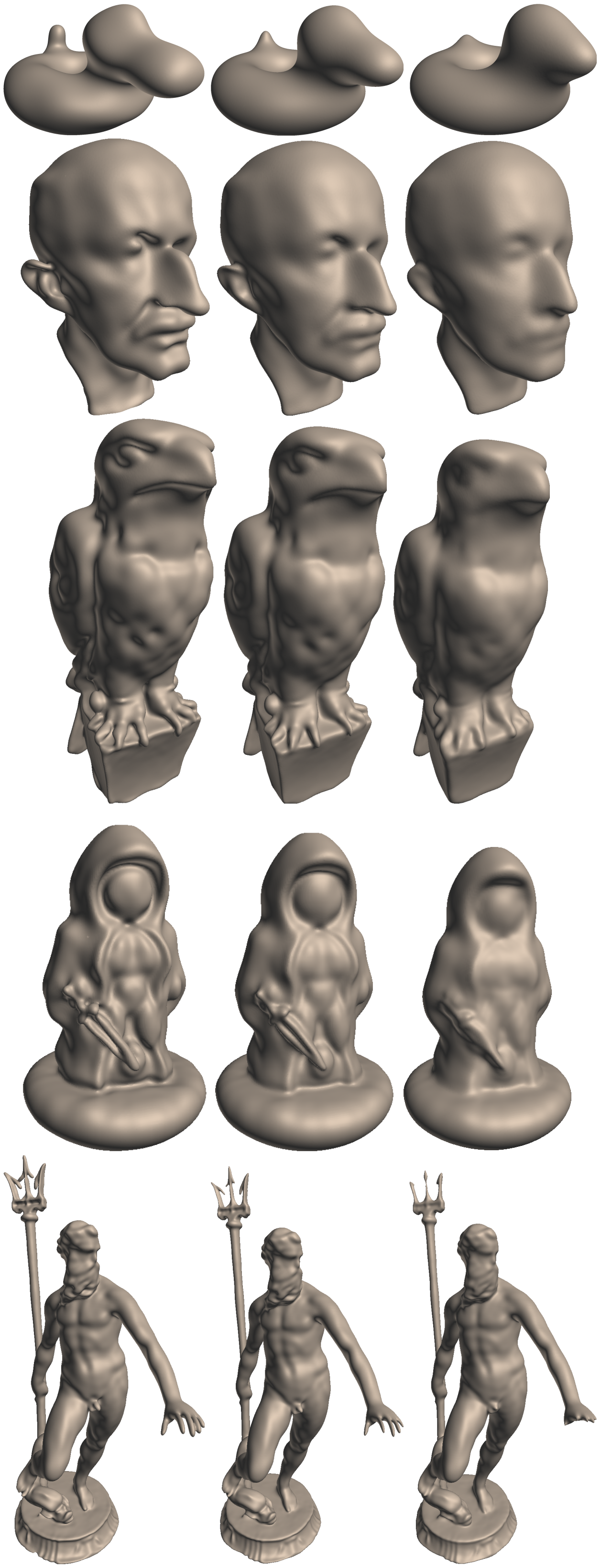

Figure 10. The zero-level sets of f_θ for Bob, Max Plank, Falcon, Witch, and Neptune models (middle). The left and right columns provide the sharpening and smoothing of the models.

4.6. Additional interpolations

Finally, Figure 4.6 shows additional examples of interpolations between neural implicit functions using the method presented in Sec 5.3 of the paper. The bracelets and chairs models are from the Thingi10K dataset [6]. We choose to interpolate the chairs because this was also an example considered by Lipschitz MLP [2]. We observed that when the features of the objects are aligned, the interpolation works like morphing between the shapes; see Line 3 of Figure 11.

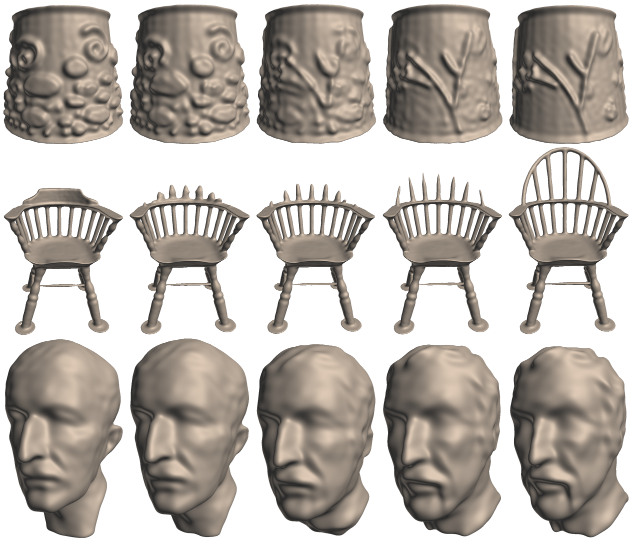

Figure 11. Interpolations between neural implicit functions. We observe that choosing surfaces with a significant overlapping of their interior regions results in better interpolations.

References

[1] Manfredo P Do Carmo. *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016. 1

[2] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via lipschitz regularization. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22. Association for Computing Machinery, 2022. 5

[3] Francisco Martín and Jesús Pérez. An introduction to the mean curvature flow. In *XXIII International Fall Workshop on Geometry and Physics, held in Granada. https://www. ugr. es/jpgarcia/investigacion. html*, 2014. 1

[4] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[5] Stephanie Wang and Albert Chern. Computing minimal surfaces with differential forms. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021. 1

[6] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 5