HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition

Supplementary Material

A. Proofs and Derivations

A.1. Proof of Proposition 1.

Here we analyze the 2D case, but the following derivation naturally generalizes to 3D. We denote $\frac{\partial u}{\partial x}$ as p_1, $\frac{\partial u}{\partial y}$ as p_2, and let $n^2(x,y) = p_1^2 + p_2^2$. Following the derivation from Kulyabov et al. [59], we can obtain the characteristic equations as follows. For any $a \neq 0$ if any field u satisfies the eikonal equation,

$$\begin{aligned} \frac{p_1}{a}\frac{\partial p_1}{\partial x} + \frac{p_2}{a}\frac{\partial p_1}{\partial y} &= \frac{n}{a}\frac{\partial n}{\partial x} \\ \frac{p_1}{a}\frac{\partial p_2}{\partial x} + \frac{p_2}{a}\frac{\partial p_2}{\partial y} &= \frac{n}{a}\frac{\partial n}{\partial y} \end{aligned} \tag{19}$$

This implies that any curve $(x(s), y(s))$ satisfying $\frac{dx}{ds} = \frac{p_1}{a}, \frac{dy}{ds} = \frac{p_2}{a}$ is the characteristic curve of p_1 and p_2. Let $a = n^2$,

$$\begin{aligned} \frac{du}{ds} &= u_x\frac{dx}{ds} + u_y\frac{dy}{ds} = p_1\frac{dx}{ds} + p_2\frac{dy}{ds} \\ &= \frac{p_1^2 + p_2^2}{a} \equiv 1 \end{aligned} \tag{20}$$

This equation holds for both u and u' when we take the characteristic curve as

$$\begin{cases} \frac{dx}{ds} = \frac{u_x}{u_x^2 + u_y^2}, \\ \frac{dy}{ds} = \frac{u_y}{u_x^2 + u_y^2} \end{cases} \tag{21}$$

We assume $s \in (0, M)$ ($M > 0$ can be infinity) is a differentiable domain such that every derivative exists and all of these equations hold. Then we prove such a parametric curve is a ray or a segment.

$$\begin{aligned} \frac{d}{ds}u_x &= u_{xx}\frac{dx}{ds} + u_{xy}\frac{dy}{ds} = u_{xx}\frac{u_x}{n} + u_{xy}\frac{u_y}{n} \\ &= \frac{1}{2n}(2u_x u_{xx} + 2u_y u_{xy}) \\ &= \frac{1}{2n}\frac{\partial(u_x^2 + u_y^2)}{\partial x} \equiv 0 \end{aligned} \tag{22}$$

Similarly, u_y is also a constant. Then the direction vector in Eq. (21) is also a constant vector. Hence, $(x(s), y(s))$ is a ray or a segment. Given $u(x_0, y_0) = u_0$, we can solve this ordinary differential equation and obtain $u(x(s), y(s)) = u_0 + s$. Respectively, $u'(x_0, y_0) = u_0 + u_{0e}$ and $u'(x(s), y(s)) = u_0 + s + u_{0e}$. On this parametic line, we have the following.

$$u'(\boldsymbol{x}(s)) - u(\boldsymbol{x}(s)) = u_{0e}. \tag{23}$$

A.2. Proposition 2 proof.

We first consider the 3D case. First, for the original solution of Eq. (4), we denote it as h. For the following disturbance equation, we denote its solution as h_e:

$$\begin{cases} (\nabla^2 - \lambda^2)h_e = 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^3 \setminus B(\boldsymbol{x_0}, \epsilon) \\ \lim\limits_{||\boldsymbol{x}|| \to \infty} h_e < \infty,\ h_e(\boldsymbol{x}) = h_{0e}\ \forall \boldsymbol{x} \in \partial B(\boldsymbol{x_0}, \epsilon) \end{cases} \tag{24}$$

Then the perturbed $h' = h_e + h$ still satisfies the screened Poisson equation in $\mathbb{R}^d \setminus (B \cup \Gamma)$. Given the spherical condition, we can analytically solve this equation by transforming it to an ordinary differential equation. Knowing that $\nabla^2 h(r) = r^{-2} \cdot d(r^2 \frac{dh}{dr})/dr$, we have:

$$\begin{aligned} &\Leftrightarrow \frac{1}{r^2}\frac{d}{dh}\left(r^2\frac{dh}{dr}\right) - \lambda^2 h = 0 \\ &\Leftrightarrow \frac{1}{r^2}\left(2r\frac{dh}{dr} + r^2\frac{d^2h}{dr^2}\right) - \lambda^2 h = 0 \\ &\Leftrightarrow \frac{d^2h}{dr^2} + \frac{2}{r}\frac{dh}{dr} - \lambda^2 h = 0 \\ &\Leftrightarrow h(r) = \frac{Ae^{-\lambda r}}{r} + \frac{Be^{\lambda r}}{r}, \quad r \geqslant \epsilon \end{aligned} \tag{25}$$

Given the boundary conditions, we obtain $A = \epsilon h_{0e} e^{\lambda\epsilon}$ and $B = 0$. Consequently, the solution is as follows.

$$h_e(r) = \frac{\epsilon}{r} h_{0e} e^{\lambda(\epsilon - r)}, \quad \forall r \geqslant \epsilon \tag{26}$$

We can further extend our analysis to 2D. Given $\nabla^2 h(r) = r^{-1} \cdot d(r\frac{dh}{dr})/dr$, we have another ODE in 2D:

$$\begin{aligned} &\Leftrightarrow \frac{d^2h}{dr^2} + \frac{1}{r}\frac{dh}{dr} - \lambda^2 h = 0 \\ &\Leftrightarrow r^2\frac{d^2h}{dr^2} + r\frac{dh}{dr} - \lambda^2 r^2 h = 0 \\ &\Leftrightarrow z^2\frac{d^2h}{dz^2} + z\frac{dh}{dz} - z^2 h = 0, \quad z = \lambda r \end{aligned} \tag{27}$$

Then we find it becomes a modified Bessel function with the following general solution:

$$h(z) = AK_0(z) + BI_0(z), \tag{28}$$

where K_0 represents the modified Bessel function of the second kind and I_0 represents the modified Bessel function of the first kind. Given the boundary conditions, we obtain $A = \frac{h_{0\epsilon}}{K_0(\lambda\epsilon)}$ and $B = 0$. Then we place r back and have the final solution as follows:

$$h(r) = \frac{h_{0\epsilon}}{K_0(\lambda\epsilon)} K_0(\lambda r) \tag{29}$$

It should be noted that K_0 converges to 0 when λr goes to infinity. More rigorously, $K_0(\lambda r)$ and $e^{-\lambda r}/\sqrt{\lambda r}$ are infinitesimals of the same order [60]. Related property then becomes similar to the 3D case.

A.3. Convergence speed proof.

Here we consider the 3D case. For single point $\boldsymbol{x}$, given the following screened Poisson equation

$$\begin{cases} (\nabla^2 - \lambda^2)h = 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^3 \setminus B(\boldsymbol{x_0}, \epsilon) \\ \lim\limits_{||\boldsymbol{x}|| \to \infty} h = 0,\ h(\boldsymbol{x}) = e^{-\lambda\epsilon}\ \forall \boldsymbol{x} \in \partial B(\boldsymbol{x_0}, \epsilon) \end{cases} \tag{30}$$

Its solution is similar to Eq. (26):

$$h(r) = \frac{\epsilon}{r} e^{-\lambda r}, \quad \forall r \geqslant \epsilon \tag{31}$$

Then we denote $S_i = \partial B(\boldsymbol{x}_i, \epsilon)$, h_i is the corresponding single point solution, r_0 is the minimum distance for any two points from Γ. When $\epsilon < r_0$ and λ is large enough, we can assume that when $i \neq j$, $h_j(S_i) = h_j(\boldsymbol{x}_i)$. Given a finite set of boundary points $\Gamma = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ that is the input of the signed distance function reconstruction task, we can use a linear combination of single point solutions to obtain a new function satisfying the screened Poisson equation and a new boundary condition that is:

$$\begin{cases} (\nabla^2 - \lambda^2)h = 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^3 \setminus \cup_{\forall i} B(\boldsymbol{x_i}, \epsilon) \\ \lim\limits_{||\boldsymbol{x}|| \to \infty} h = 0, \quad h(\boldsymbol{x}) = e^{-\lambda\epsilon} \quad \forall \boldsymbol{x} \in \cup_{\forall i} S_i \end{cases} \tag{32}$$

We then set the boundary condition equation to solve the coefficients:

$$\begin{pmatrix} h_1(S_1) & h_2(S_1) & \cdots & h_N(S_1) \\ h_1(S_2) & h_2(S_2) & \cdots & h_N(S_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(S_N) & h_2(S_N) & \cdots & h_N(S_N) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} e^{-\lambda\epsilon} \\ e^{-\lambda\epsilon} \\ \vdots \\ e^{-\lambda\epsilon} \end{pmatrix} \tag{33}$$

We denote the matrix here as $H \in \mathbb{R}^{N \times N}$. Its every diagonal element is $e^{-\lambda\epsilon}$, which is the largest in that row. This equation has at least one solution $C \in \mathbb{R}^N$ and every $c_i \in (0, 1)$. Finally, $h = (h_1, \ldots, h_N)C$ is the solution of Eq. (32). If we set $r_i = ||\boldsymbol{x} - \boldsymbol{x}_i||_2$, then

$$h(\boldsymbol{x}) = \epsilon \sum_{i=1}^{n} c_i \frac{e^{-\lambda r_i}}{r_i} \tag{34}$$

Now we analyze the bound of $\frac{1}{\lambda}\ln(h_\lambda(\boldsymbol{x})) + d_\Gamma(\boldsymbol{x})$. First, we prove its upper bound is $\frac{1}{\lambda}[\ln\frac{\epsilon}{d_\Gamma(\boldsymbol{x})} + \ln(N)]$.

$$\begin{aligned} \frac{1}{\lambda}\ln h + d_\Gamma &= \frac{1}{\lambda}\left(\ln\epsilon + \ln\sum_{i=1}^{n} c_i \frac{e^{-\lambda r_i}}{r_i}\right) + d_\Gamma \\ &\leqslant \frac{1}{\lambda}\left(\ln\epsilon + \ln\frac{e^{-\lambda d_\Gamma}}{d_\Gamma} + \ln\sum_{i=1}^{N} c_i\right) + d_\Gamma \\ &\leqslant \frac{1}{\lambda}\left(\ln\frac{\epsilon}{d_\Gamma} + \ln N\right) \end{aligned} \tag{35}$$

Here, the first inequality holds because $d_\Gamma \leqslant r_i$ for all i and every $h_i(r)$ is monotonically decreasing w.r.t. r. Then every $c_i < 1$, so we can obtain the second inequality.

Second, we prove its lower bound is $\frac{1}{\lambda}\ln\frac{\epsilon}{d_\Gamma(\boldsymbol{x})}$.

For any row in Eq. (33), since we know that the largest element is on the diagonal equal to $e^{-\lambda\epsilon}$, we have

$$e^{-\lambda\epsilon} = \sum_{i=1}^{N} c_i h_i(S_j) \leqslant \sum_{i=1}^{N} c_i h_j(S_j) = e^{-\lambda\epsilon}\sum_{i=1}^{N} c_i \tag{36}$$

Hence, $\sum_{i=1}^{N} c_i \geqslant 1$. Next, we can set $h_0(r_i) = h_i(\boldsymbol{x})$, where h_0 is the decay pattern in Eq. (31). h_0 is a convex and monotonically decreasing function so that we can apply these inequalities.

$$\begin{aligned} \frac{1}{\sum_{i=1}^{N} c_i} h(\boldsymbol{x}) &= \frac{\sum_{i=1}^{N} c_i h_0(r_i)}{\sum_{i=1}^{N} c_i} \\ &\geqslant h_0\left(\frac{\sum_{i=1}^{N} c_i r_i}{\sum_{i=1}^{N} c_i}\right) \geqslant h_0(d_\Gamma(\boldsymbol{x})) \end{aligned} \tag{37}$$

Given $\sum_{i=1}^{N} c_i \geqslant 1$, we have $h(\boldsymbol{x}) \geqslant \epsilon\frac{e^{-\lambda d_\Gamma}}{d_\Gamma}$. By basic transformations of this, we obtain the whole inequality as follows:

$$\frac{1}{\lambda}\ln\frac{\epsilon}{d_\Gamma} \leqslant d_\Gamma - |u_\lambda| \leqslant \frac{1}{\lambda}\left(\ln\frac{\epsilon}{d_\Gamma} + \ln N\right) \tag{38}$$

where $|u_\lambda| = -\frac{1}{\lambda}\ln h_\lambda$. It should also be noted that $\ln\frac{\epsilon}{d_\Gamma(\boldsymbol{x})}$ is a constant scalar field so that when $\boldsymbol{x}$ is fixed and λ increases to infinity, $d_\Gamma - |u_\lambda|$ is first-order infinitesimal.

B. Experiments

In all of our experiments, we use $p = 1$ for Eq. (2) and Eq. (3).

B.1. 1D Verification

We can easily identify which loss function is not a sufficient constraint by observing its input. As discussed in the main text, if a loss function depends solely on the first-order derivative of the SDF or any higher-order derivatives, it is inherently insufficient as a constraint and cannot exclude other non-SDF solutions.

Our experiments also demonstrate that implementations optimizing SDFs with the closest point [2, 51, 52] may struggle when making queries near the interpolated surface. As mentioned at the end of Section 3, after the neural network interpolates these points due to spectral bias [58], our loss remains effective in faithfully capturing the distance to the interpolated surface.

For other cases, we can easily test whether a constraint is insufficient with simple 1D verifications.

We found one possible candidate to be a sufficient constraint from Marschner et al. [5] with following loss:

$$L_{\text{CP}} = \int_{\Omega} |u[\boldsymbol{x} - u(\boldsymbol{x})\nabla u(\boldsymbol{x})]|^2 d\boldsymbol{x} \tag{39}$$

The loss is designed to optimize constructive solid geometries and formulated using both neural network outputs and their gradients, ensuring that when taking a step in the direction indicated by the SDF, the new position should have a distance value close to zero.

However, our experiment in Fig. 11 reveals that it cannot exclude non-SDF solutions like $u(\boldsymbol{x}) \equiv 0$, confirming that it is not a sufficient constraint.

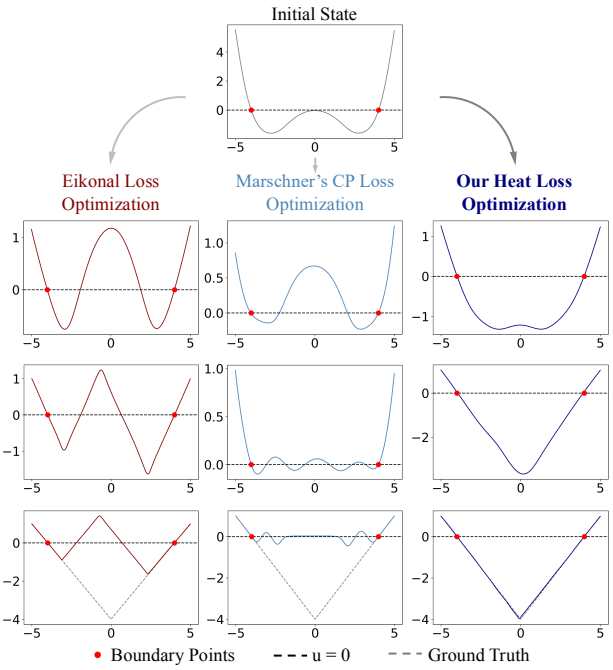


Figure 11. Marschner et al. [5] proposed a CP loss which is a possible candidate to be a sufficient constraint. However, our 1D experiment illustrates that it is incapable of converging to the actual signed distance function (dashed line), even when the output minimizes this loss almost everywhere. The x axis is the domain and the y axis shows the output of the implicit function. The middle two rows display the intermediate states of the optimization, while the bottom row presents the final results.

B.2. 2D Dataset

Fig. 12 provides an overview of the ground truths in the 2D dataset. Starting with simple shapes from DiGS [3] and StEik [4], we extended the dataset to include 14 shapes. We generate a total of 150,000 points along the boundaries in a single vector image. For each iteration, we randomly select 10% of the generated points to compute the boundary loss.

In our first 2D experiments (Table 6), we adopt the original hyperparameters and settings from DiGS [3] and StEik [4] as our baselines, making only one modification: extending the training iterations from 10k to 20k to better learn complex shapes.

In our setup, we compute the heat loss using the importance sampling method, which combines a mixture distribution of 1:1 uniform samples in $[-1.5, 1.5]^2$ and Gaussian samples with an isotropic $\sigma = 0.5$. For each iteration, we use 4,096 points for both the uniform and Gaussian samples, whereas DiGS and StEik generate 15,000 points to compute their derivative-based loss. We fix λ in the heat loss while employing two schedulers: one for the heat loss and another for the eikonal loss. Towards the end of training, the heat loss is gradually weakened, and the eikonal loss is strengthened. This strategy aligns with the approach used in DiGS and StEik. Our results are presented in Fig. 12 as well. However, as shown in the ablation study (Table 2), removing the scheduler does not result in significant differences.

We also visualize the outcomes of the baseline methods in Fig. 13. In the ablation study, the boundary loss coefficient remains constant and identical across all experiments, and the same scheduler is applied to the eikonal loss. We also adopted the original loss weight ratios from the experiments of DiGS [3] in the fourth column and StEik [4] in the seventh column. All losses, except for the eikonal loss, are applied without a scheduler. The outputs of the original DiGS and StEik models are also shown in the fifth and eighth columns, respectively.

All other models make some errors in topology. Even when their topologies are correct, details like the Target and House shapes in the first two columns are missing. When

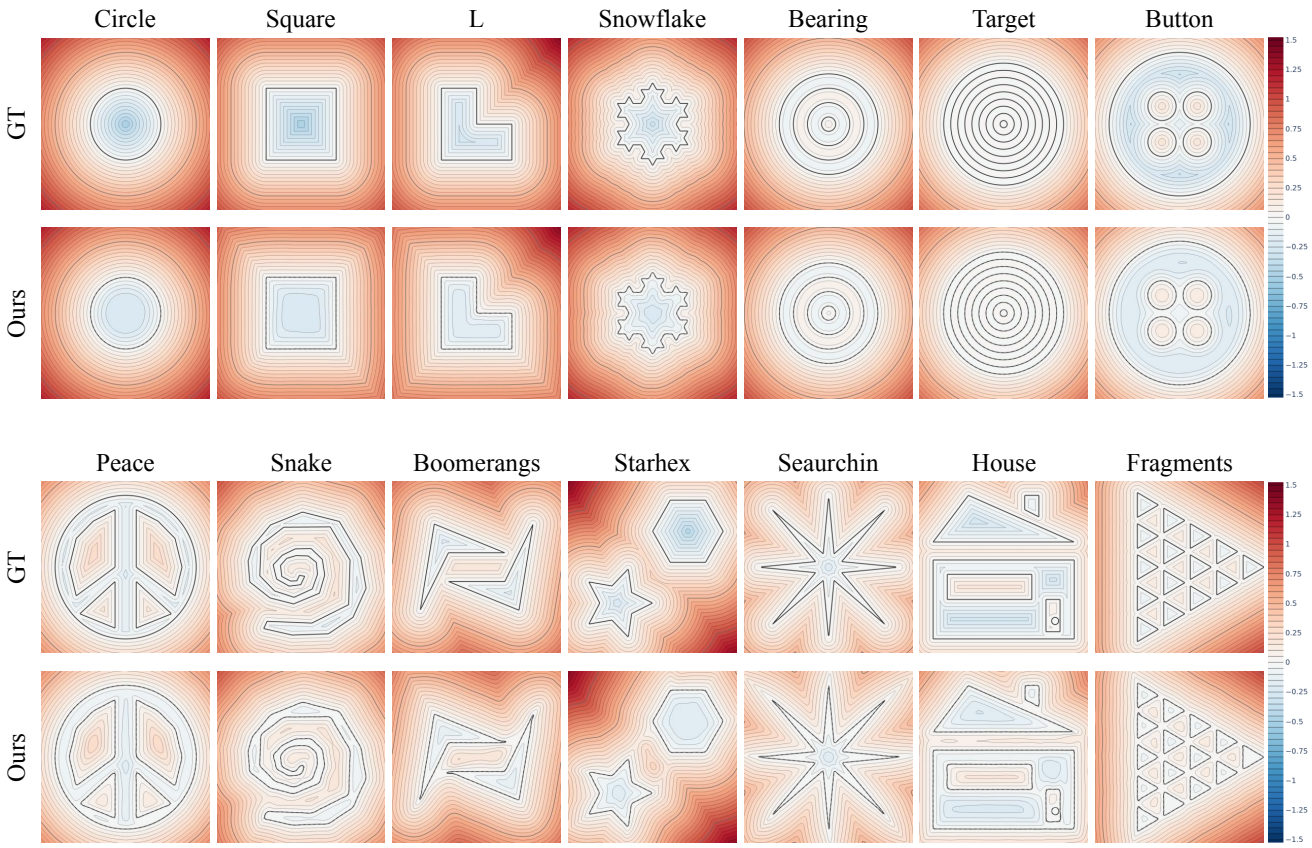


Figure 12. This figure illustrates all 14 vector shapes in our 2D dataset. Each visualization lies within the range $[-1.2, 1.2]^2$. Spanning a spectrum from simple to complex, the dataset encompasses various topologies, smooth and irregular boundaries, as well as configurations with single and multiple objects. Our HOTSPOT model accurately reconstructs all curves while preserving their correct topologies.

using a relatively smaller learning rate in the ablation study for DiGS and StEik, the outputs become overly flat, despite maintaining the same ratio among the loss weights from their paper. This can be interpreted as a limitation of their derivative-based losses, which, as a corollary of the eikonal equation, only serve as a necessary condition for the equation, encouraging the condition $|\nabla u| = c$, where c can be any constant. With a small learning rate, their losses trap the outputs at $|\nabla u| = 0$.

In our framework, we analyze the influence of different values of λ and various coefficients of the eikonal loss, with visualizations presented in Fig. 14. This experiment replicates the settings from the ablation study, except for the values of λ, w_e, and the iteration number, with all schedulers removed. To ensure full convergence, we extend the training iterations from 20k to 200k.

From Fig. 4 and Fig. 14, we observe the influence of different λ. When λ is very small, heat from the boundaries diffuses to distant regions, causing $h \approx 1$ almost everywhere in the test region. As a result, $u \approx 0$ across the domain, leading to an overly flat signed distance function with many extra boundaries. As λ increases, the outputs become more regular. Notably, even without the eikonal loss, setting $\lambda = 10$ yields outputs that are more regular than several baselines in Fig. 13. However, as λ continues to increase, the factor $e^{-\lambda|u|}$ in the loss computation diminishes rapidly, especially where $|u|$ is large initially or grows during training. This makes optimization without the eikonal loss increasingly challenging and less effective. For instance, in the $\lambda = 50$ and $\lambda = 100$ subfigures with $w_e = 0$, the values in the upper-right region remain greater than 1.0 even after 200k iterations. Although our neural network provides some output values in these regions, they are significantly larger than the ground truth.

Incorporating the eikonal loss stabilizes the training process and promotes a more regular field. When λ is small, the approximation from Eq. (5) is weakly achieved, but the eikonal loss helps regulate the output and prevents it from becoming overly flat. When λ is large, the eikonal loss dominates in regions where $\lambda|u|$ is substantial. However, if the eikonal loss is overly strong, extra boundaries and local optima may re-emerge.

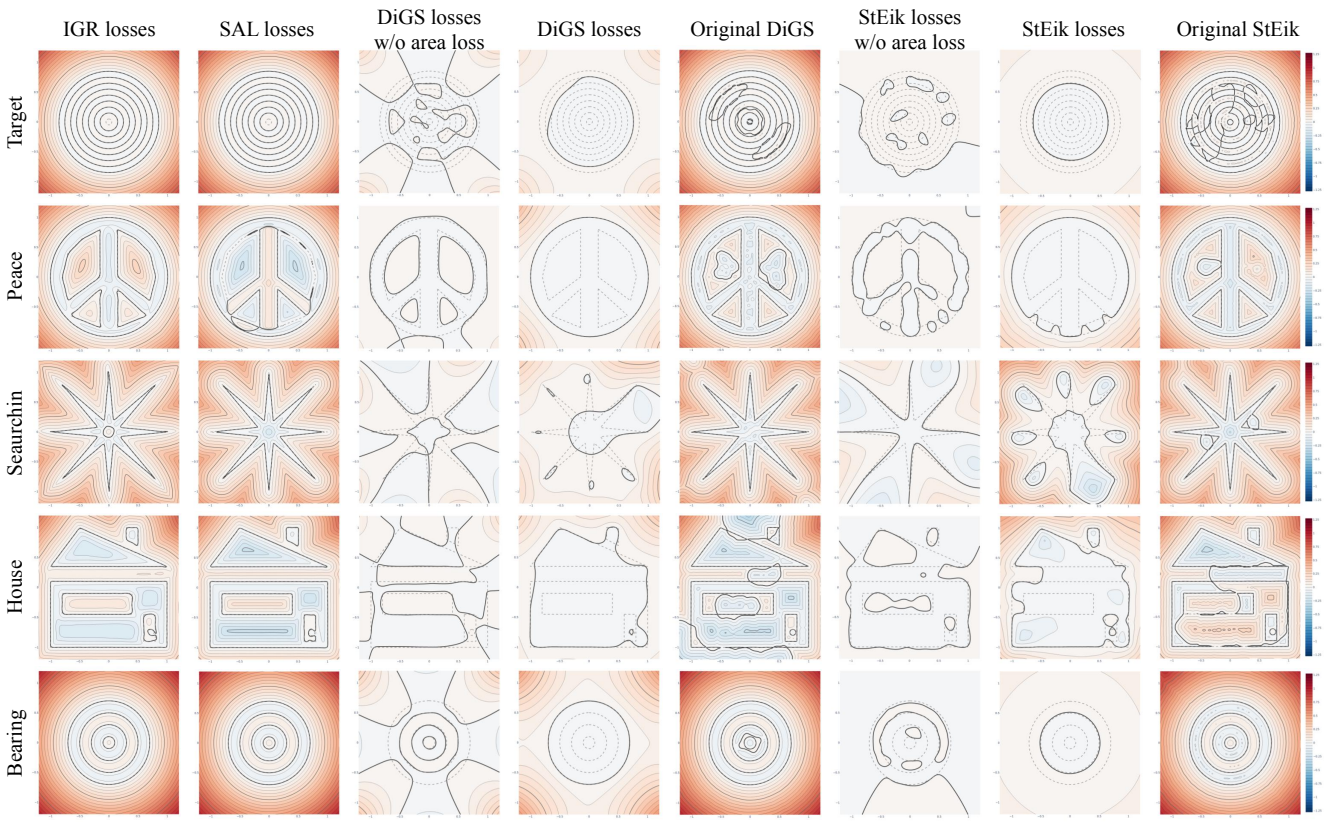


Figure 13. This figure demonstrates how the baselines fail with complex shapes. The difference between the fourth and fifth columns (from left to right), as well as between the seventh and eighth columns, is that the former one is generated in the ablation study, while the other is generated by the original model. Dashed lines represent the true boundaries, while the boldest line indicates the reconstructed boundaries.

This does not mean that users must carefully balance the hyperparameters w_e, w_h, and λ. We have found effective ways to choose them. In our subsequent experiments, the schedulers for λ and w_e ensure robust shaping capabilities across various shapes and distance ranges in the ShapeNet dataset [55]. Mimicking a real annealing process, we gradually increase λ and w_e, allowing the heat loss to shape and stretch most regions first, helping the optimization escape local optima. As the heat field cools due to the increasing absorption coefficient λ, the eikonal loss maintains the stretching and assumes control in remote regions.

B.3. ShapeNet

We show full metrics for the ShapeNet dataset in Tables 7, 8, and 9. We compare our method with the state-of-the-art methods, including SAL [2], SIREN without normalization [34], Neural-Singular-Hessian [45], Neural-Pull [51], DiGS [3], and StEik [4]. Our method outperforms all other methods in terms of surface reconstruction metrics, including IoU, Chamfer distance, and Hausdorff distance. In terms of distance query metrics, our method achieves near-top performance across RMSE, MAE, and SMAPE.

Notably, in near-surface regions, our approach outperforms all other methods, including SAL [2] and Neural-Singular-Hessian [45], across these metrics. As discussed in the main text, while approximating the SDF using the closest point information may be reasonably accurate for distant regions, it proves inadequate for near-surface regions. This limitation is particularly critical for sphere tracing—one of the most important applications of SDF—as it heavily relies on accurate field values in near-surface regions due to the high density of queries in these areas.

In contrast, as we introduced at the end of Section 3, after the neural network interpolates these points due to spectral bias [58], our loss remains effective in faithfully capturing the distance to the interpolated surface. Our experiments on sphere tracing further substantiate this claim, demonstrating the infeasibility of approximating the SDF using only the closest point information and highlighting the superiority of our proposed model.

Our test region is defined in the same way as in DiGS and StEik. Both methods rescaled the circumscribed sphere centered at the geometric center of a point cloud to a unit sphere and designated the circumscribed cube $[-1, 1]^3$ as

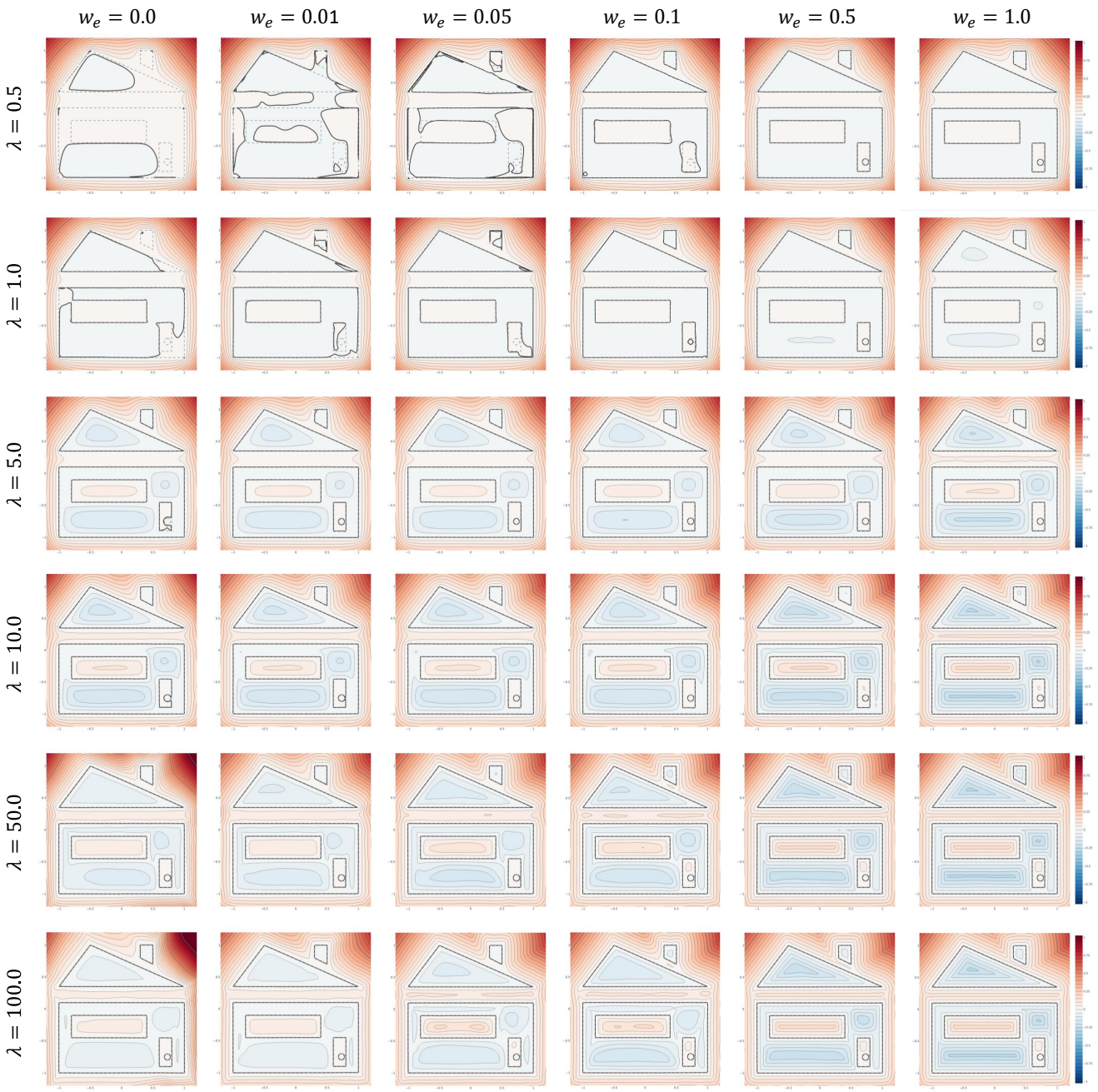


Figure 14. Signed distance function reconstructions of the vector image House with varying values of λ and w_e.

	IoU ↑			Chamfer Distance ↓			Hausdorff Distance ↓		
	mean	median	std	mean	median	std	mean	median	std
DiGS [3]	0.7882	0.9359	0.2803	0.0055	0.0037	0.0046	0.1267	0.1350	0.1088
StEik [4]	0.6620	0.7305	0.3224	0.0073	0.0051	0.0068	0.1425	0.1654	0.1146
Ours	**0.9870**	**0.9888**	**0.0083**	**0.0014**	**0.0013**	**0.0003**	**0.0153**	**0.0150**	**0.0100**

Table 5. Comparison of 2D dataset reconstruction metrics.

	RMSE ↓			MAE ↓			SMAPE ↓		
	mean	median	std	mean	median	std	mean	median	std
DiGS [3]	0.0597	0.0504	0.0511	0.0315	0.0253	0.0351	0.3363	0.2355	0.3414
StEik [4]	0.0725	0.0335	0.0903	0.0419	0.0108	0.0574	0.4222	0.2223	0.4409
Ours	**0.0199**	**0.0189**	**0.0130**	**0.0101**	**0.0072**	**0.0060**	**0.0699**	**0.0693**	**0.0226**

Table 6. Comparison of 2D dataset distance queries.

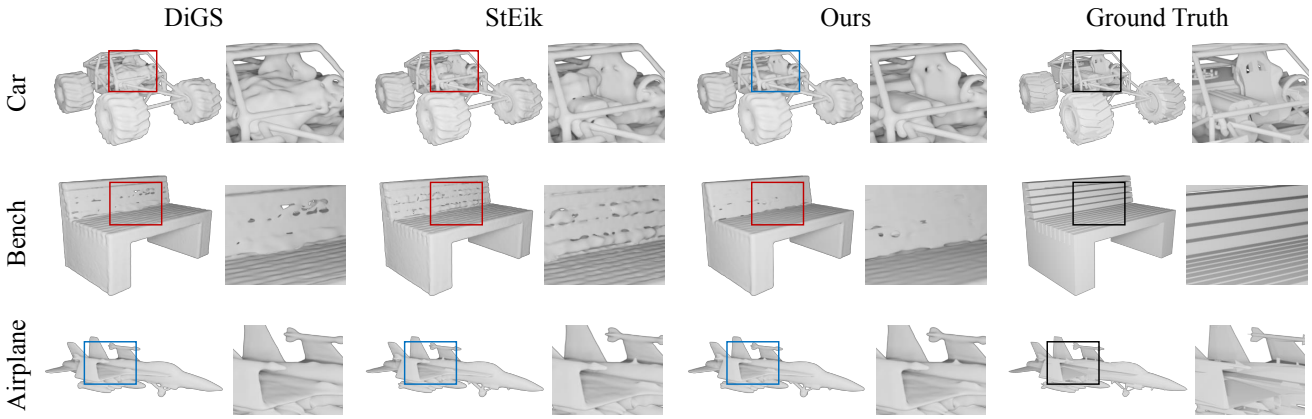


Figure 15. Additional results from the ShapeNet dataset [55]. We present the bench, where none of the methods achieves a satisfactory surface reconstruction, and the airplane, where all methods successfully capture detailed structures.

	IoU ↑			Chamfer Distance ↓			Hausdorff Distance ↓		
	mean	median	std	mean	median	std	mean	median	std
SAL [2]	0.7400	0.7796	0.2231	0.0074	0.0065	0.0048	0.0851	0.0732	0.0590
SIREN wo/ n [34]	0.4874	0.4832	0.4030	0.0051	0.0038	0.0036	0.0558	0.0408	0.0511
NSH [45]	0.7293	0.9285	0.3538	0.0036	0.0033	0.0017	**0.0324**	0.0231	**0.0286**
Neural-Pull [51]	0.7300	0.7972	0.2229	0.0114	0.0073	0.0161	0.1334	0.0840	0.1287
DiGS [3]	0.9636	0.9831	**0.0903**	**0.0031**	**0.0028**	**0.0016**	0.0435	**0.0168**	0.0590
StEik [4]	**0.9641**	**0.9848**	0.1052	0.0032	**0.0028**	0.0028	0.0368	0.0172	0.0552
Ours	**0.9796**	**0.9842**	**0.0203**	**0.0029**	**0.0028**	**0.0012**	**0.0250**	**0.0153**	**0.0360**

Table 7. Surface reconstruction metrics on ShapeNet [55]. Bold and underlined data: optimal; bold only: suboptimal. Same below.

	RMSE ↓			MAE ↓			SMAPE ↓		
	mean	median	std	mean	median	std	mean	median	std
SAL [2]	**0.0251**	**0.0197**	0.0270	**0.0142**	**0.0116**	0.0108	0.1344	0.1064	0.1032
SIREN wo/ n [34]	0.5009	0.4842	0.1769	0.4261	0.4027	0.1811	1.2694	0.9859	0.5195
NSH [45]	0.3486	0.2469	0.2566	0.2891	0.1780	0.2508	0.7386	0.4897	0.5424
Neural-Pull [51]	**0.0093**	**0.0067**	**0.0121**	**0.0060**	**0.0053**	**0.0042**	**0.0673**	**0.0505**	**0.0612**
DiGS [3]	0.1194	0.1107	0.0597	0.0725	0.0644	0.0423	0.2140	0.2162	0.0935
StEik [4]	0.0387	0.0338	0.0229	0.0248	0.0222	0.0142	0.0931	0.0843	0.0748
Ours	0.0281	0.0259	**0.0136**	0.0176	0.0160	**0.0082**	**0.0540**	**0.0514**	**0.0243**

Table 8. Overall distance query metrics on ShapeNet [55].

their test region. We evaluate our results within the exact same coordinate system. During training, however, we rescale the point cloud to achieve an adaptive λ described in the main text.

To compute the IoU and ground truth distances, we utilized the Occupancy Network [16] and a point cloud completion model [57]. A dense grid was generated within $[-1, 1]^3$ to evaluate the metrics. For near-surface queries,

	RMSE near surface ↓			MAE near surface ↓			SMAPE near surface ↓		
	mean	median	std	mean	median	std	mean	median	std
SAL [2]	0.0245	0.0252	0.0075	0.0182	0.0189	0.0059	0.6848	0.6890	0.2488
SIREN wo/ n [34]	0.0513	0.0401	0.0404	0.0382	0.0206	0.0351	0.8858	0.5406	0.7483
NSH [45]	0.0876	0.0798	0.0383	0.0686	0.0601	0.0342	0.8830	0.7254	0.4744
Neural-Pull [51]	**0.0123**	**0.0098**	0.0088	0.0087	0.0076	0.0047	0.3856	0.3459	0.1439
DiGS [3]	0.0152	0.0135	0.0081	**0.0081**	**0.0074**	**0.0037**	**0.1760**	**0.1657**	**0.0660**
StEik [4]	0.0147	0.0130	**0.0070**	**0.0081**	**0.0074**	0.0041	0.1770	0.1664	0.0859
Ours	**0.0094**	**0.0078**	**0.0049**	**0.0047**	**0.0042**	**0.0020**	**0.1206**	**0.1163**	**0.0313**

Table 9. Distance function query metrics for near surface region on ShapeNet [55].

we filtered points with a ground truth distance smaller than 0.1 to compute the accuracy, as presented in Table 9. Our method demonstrates a remarkable lead, reducing losses by more than one-third compared to the second-best model.

To illustrate the improved quality of our level sets, we also provide sectional views from different models for comparison in Fig. 16.

Our level set near the surface is smoother and more regular, offering significant advantages for downstream tasks such as sphere tracing.

B.4. Complex Topology Reconstruction

We adopt five high-genus geometries from Mehta et al. [1] and generate 3D point clouds for them, including Bunny, Genus6, VSphere, Dino, and Kangaroo. Additionally, we use the mesh of VSphere to create bilayer and trilayer VSphere, resulting in a total of seven shapes with complex topologies. We compare our method with SAL [2], DiGS [3], and StEik [4] on these shapes, presenting the visual results in Fig. 1, Fig. 8, Fig. 17, and Fig. 18. Our method runs for 10k iterations. To ensure sufficient convergence and minimize extra boundaries, we run the other methods for 20k iterations on Bunny, Genus6, VSphere, Dino, and Kangaroo, and for 100k iterations on the bilayer and trilayer VSphere. Despite the increased iterations, the other methods fail to reconstruct the correct topology and generate extra boundaries, whereas our method successfully reconstructs the correct topology for all shapes.

B.5. Surface Reconstruction Benchmark (SRB)

SRB consists of 5 noisy scans, each containing point cloud and normal data. We compare our method against the current state-of-the-art methods on this benchmark without using the normal data. The results are presented in Table 10, where we report the Chamfer (d_C) and Hausdorff (d_H) distances between the reconstructed meshes and the ground truth meshes.

Additionally, we provide the corresponding one-sided distances ($d_{\vec{C}}$ and $d_{\vec{H}}$) between the reconstructed meshes and the input noisy point cloud. It is worth noting that one-sided distances are used here to maintain consistency with the historical choice of previous methods.

Our improvement is less pronounced compared to prior methods, as the SRB dataset represents a relatively simple benchmark without complex structures. Additional visual results are provided in Fig. 19.

Compare with	GT		Scans	
Method	d_C	d_H	$d_{\vec{C}}$	$d_{\vec{H}}$
IGR wo n	1.38	16.33	0.25	2.96
SIREN wo n	0.42	7.67	**0.08**	**1.42**
SAL [2]	0.36	7.47	0.13	3.50
IGR+FF [49]	0.96	11.06	0.32	4.75
PHASE+FF [49]	0.22	4.96	**0.07**	1.56
DiGS [3]	**0.19**	3.52	**0.08**	1.47
StEik [4]	**0.18**	**2.80**	0.10	1.45
Ours	**0.19**	**3.17**	0.09	**1.36**

Table 10. Surface reconstruction metrics on SRB [56].

B.6. Sphere Tracing

For implementation details, we adopt the sphere tracing algorithm [21], as implemented in Yariv et al.'s work [39]. For each pixel, the algorithm advances along the ray by the signed distance function value at the current point, repeating this process until one of the following conditions is satisfied: *convergence*, where the SDF value falls below a threshold of 5.0×10^{-5}; *divergence*, where the ray steps outside the unit sphere; or the maximum step limit of 30 is reached.

For signed distance functions, we use trained models from SAL [2], DiGS [3], StEik [4], and our proposed method. The evaluation is conducted on five randomly selected objects (airplane, car, watercraft, rifle, and lamp) from ShapeNet [55]. For each object, we generate ten camera poses arranged in a circular trajectory around the central object, with a radius of 1.0 and a height of 0.5. The rendered images have a resolution of 500×500 pixels.

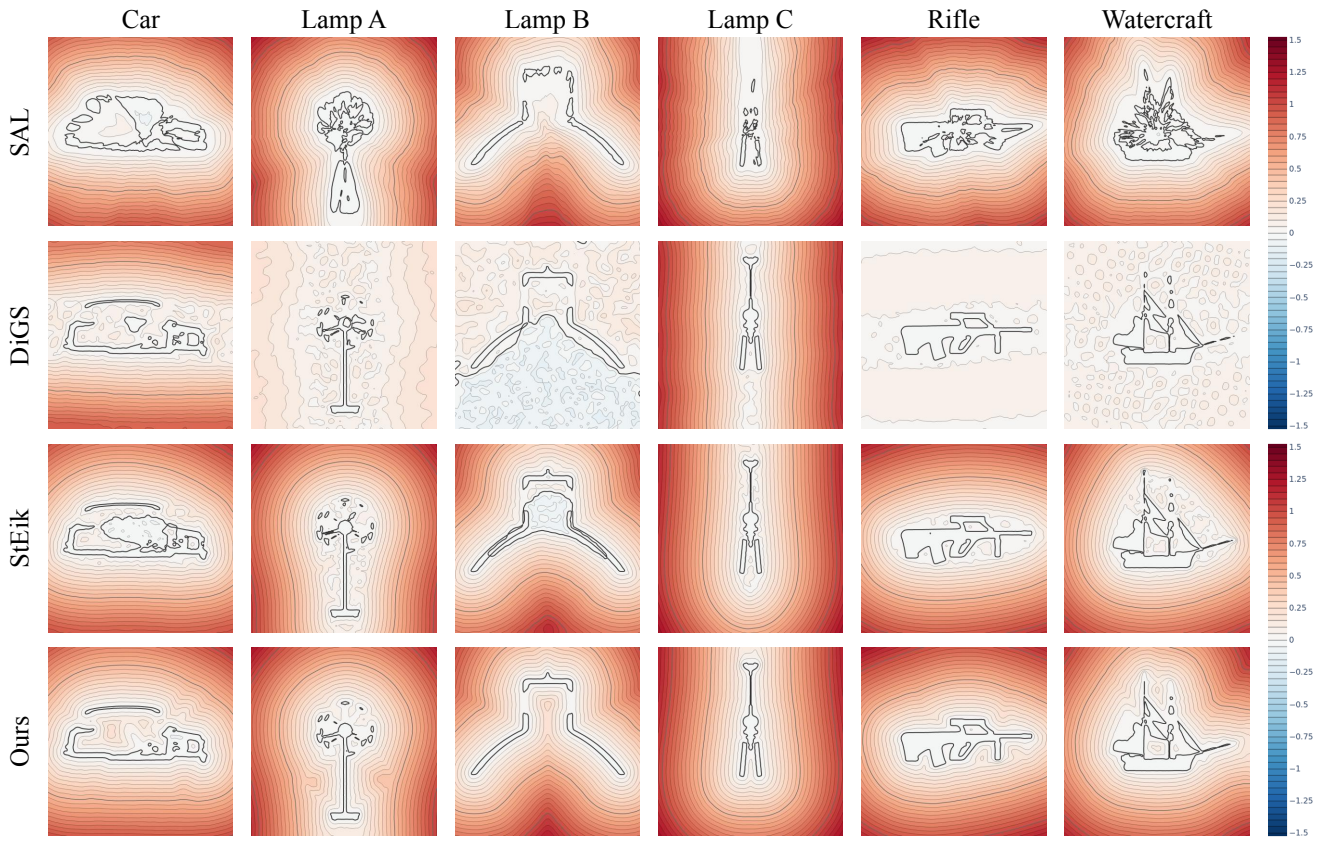


Figure 16. Sectional views of ShapeNet [55] signed distance function reconstruction results. Car, Lamp A, and Lamp B are shown in Fig. 9, while Lamp C, Rifle, and Watercraft are presented in Fig. 10 and Fig. 20. By cross-verifying with them, HOTSPOT achieves reconstructions with fewer extra boundaries, more regularized level sets, and accurate topologies.

Fig. 10 and Fig. 20 illustrate the number of steps required for each pixel until ray marching terminates under one of the three conditions described above. Brighter pixels correspond to rays that are harder to converge, requiring more queries, whereas our model produces relatively darker results compared to other models. Furthermore, the histograms for our model are more skewed to the left, highlighting its efficiency. These observations demonstrate that our model excels at early divergence detection in non-intersected regions and requires fewer steps to locate the surface in intersected regions. This efficiency stems from the smoothness and accuracy of our signed distance function, particularly its high quality near the surface, which significantly enhances the rendering performance of sphere tracing.

Because Fig. 10 and Fig. 20 only visualize the computation costs, to verify the accuracy of the object shapes in rendering, we visualize the depth and surface normals at the intersections. For rays that do not converge within 30 iterations, the intersection is approximated by identifying sign transitions at 100 equally spaced sampling points along the ray. The normal vector at the intersection, denoted by $\hat{x}$, is computed using Eq. (40) to enhance the visualization of the geometries. Non-intersected areas are masked in white. Fig. 21 demonstrates that our model accurately detects and represents the object's surface in this downstream application, while other models show artifacts and distortions.

$$\hat{n}(\theta) = \frac{\nabla f(\hat{x}(\theta), \theta)}{\|\nabla f(\hat{x}(\theta), \theta)\|_2}. \tag{40}$$

These results demonstrate that in sphere tracing rendering, the distance query accuracy of our model effectively guides the ray toward the object's surface, enhancing rendering efficiency. Additionally, its precision at the zero level set ensures an accurate representation of the object.

C. Relation to PHASE [49]

While derived from very different mathematical principles and a different motivation, our method turns out to have a close relation to the PHASE model proposed by Lipman. The PHASE model essentially simulates two fluids finding equilibrium in a container by minimizing an energy func-

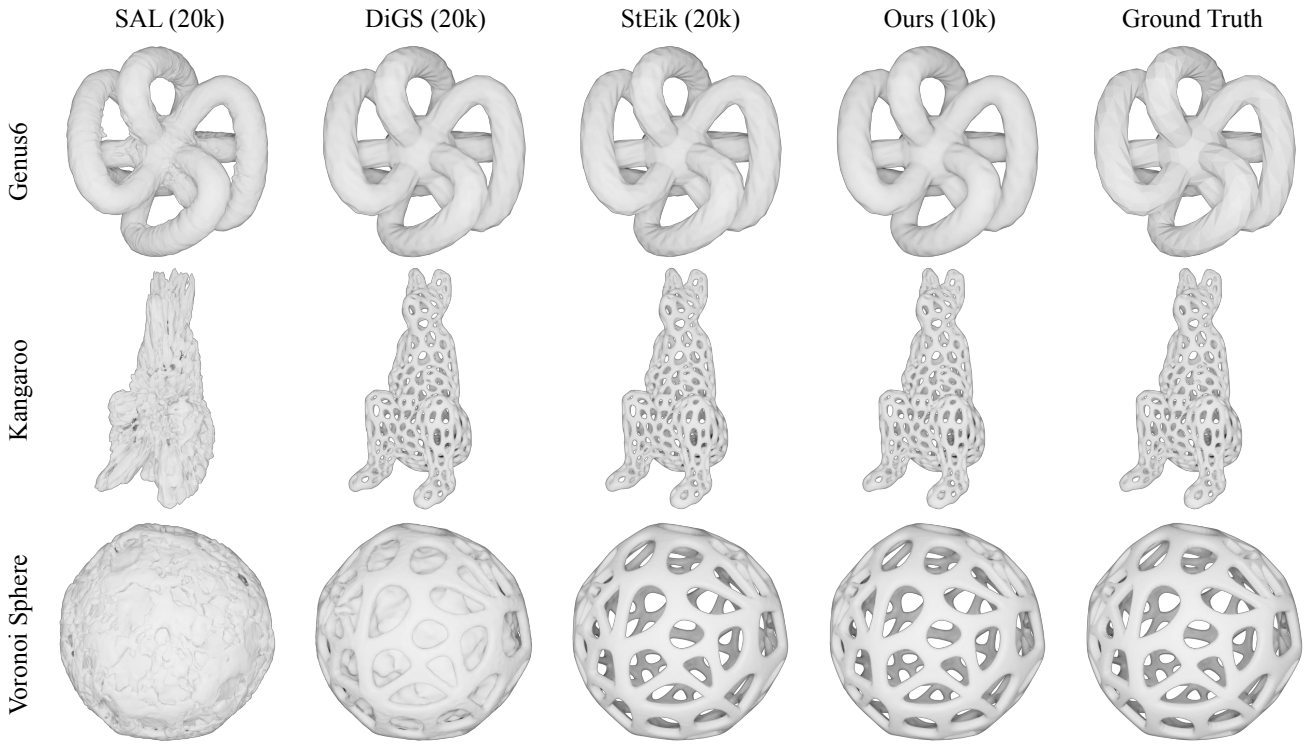


Figure 17. Visualizations of Genus6, Kangaroo, and Voronoi Sphere from Mehta et al [1]. We show the number of iterations used in training in the parentheses. We achieve excellent results with only half the iterations, successfully capturing the correct topologies.

tional. A smooth approximation of the indicator of different fluids is involved and interpreted as an *occupancy function* $o(\boldsymbol{x})$ such that o outputs 1 when outside of the object, -1 when inside the object, and 0 when exactly on the surface. Lipman further adds a reconstruction loss to encourage $o(\boldsymbol{x})$ to vanish at the boundary and adapts the Van der Waals-Cahn-Hilliard theory of phase transitions [61, 62], resulting in the following functional to minimize:

$$\mathcal{F}(o) = w_b \mathcal{L}(o) + \int_\Omega \epsilon \|\nabla o\|^2 + W(o), \tag{41}$$

where w_b is the weight of the reconstruction loss, $\mathcal{L}(o)$ is the reconstruction loss that encourages o to be zero at the boundary, ϵ is a small positive constant, and $W(o)$ is a potential. Lipman showed that by choosing a double-well potential $W(o) = o^2 - 2|o| + 1$, the minimizer can be converted into an approximated signed distance function s through a log transform:

$$s = -\sqrt{\epsilon} \ln(1 - |o|) \operatorname{sign}(o). \tag{42}$$

It turns out that our heat field h when optimized under our heat loss is closely related to the regularized occupancy function o. If we set $h = 1 - |o|$ and $\lambda = \epsilon^{-\frac{1}{2}}$ and solve for the screened Poisson equation (Eq. (4)), we would obtain PHASE's regularized occupancy.

However, there are several crucial differences between our approach and PHASE where our theory and derivation has led to additional insights and huge differences in implementations and results.

First, based on a theoretical framework aiming the area minimization, PHASE's theory promotes setting the weight w_b of the boundary loss (Eq. (2)) to $w_b = \epsilon^\alpha$ where $\alpha \in (\frac{1}{4}, \frac{1}{2})$. However, we found that finding the exact minimal area is detrimental to the optimization, as discussed in Section 3. On the contrary, according to our theory and experiments, we find it necessary to maintain boundaries between points and set the weight w_b to a much higher value to ensure the boundary condition of the screened Poisson equation is satisfied (Eq. (4)), as states in main text.

Theorem 2 in the PHASE paper states that to achieve minimal area, w_b should converge to 0 as ϵ approaches an infinitesimal value. However, this interpretation does not align with the nature of the signed distance function reconstruction task which should complete the manifold and connect points, as discussed at the end of Section 3, and may lead to unintended consequences as follows: When the boundary weight w_b is overly small, the signed distance function values of boundary points cannot even converge to close to 0. When the weight w_b is still not large enough, their output collapses from a surface-based-distance signed

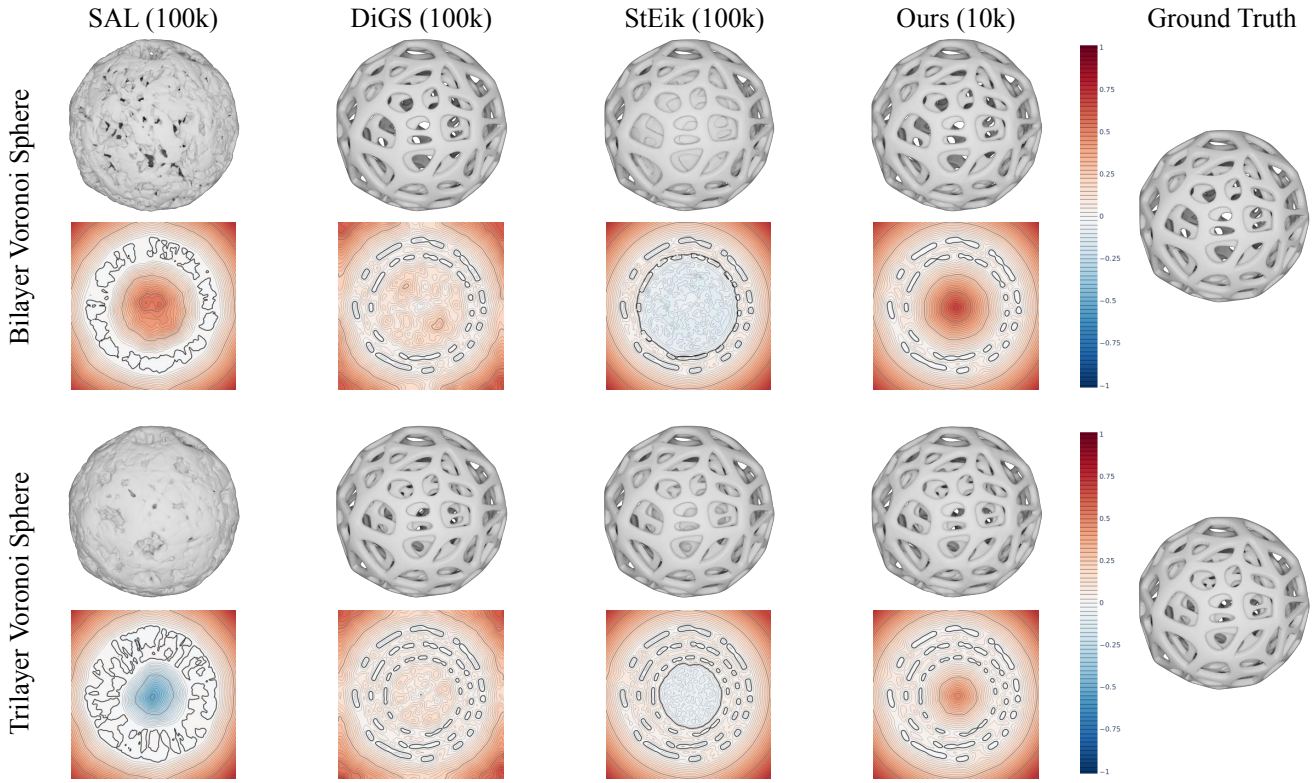


Figure 18. Visualizations of bilayer and trilayer Voronoi Spheres. Beyond achieving accurate topology reconstructions, the sectional views reveal that our level set is the only one free from chaotic and noisy interiors, ensuring meaningful representations.

Figure 19. Visualization results on the SRB dataset. From left to right: Anchor, Daratech, DC, Garagoyle, and Lord Quas.

distance function to a point-cloud-based-distance *unsigned* distance function upon reaching the target where the boundaries are only point clouds and the area becomes almost zero. In contrast, our model demonstrates robust and accurate performance, effectively connecting points and interpolating boundaries by taking advantages of neural network's spectral bias [58], with a large boundary weight w_b and an adaptive absorption λ, as discussed in Section 6.

In addition to Fig. 6, we show more empirical results on our 2D dataset supporting our claim here in the supplementary. In Table 11, we show the visual results of PHASE on a single circle across different boundary weight choices, and compute the surface reconstruction and distance metrics. We show visual results on the rest of the 2D dataset in Fig. 23, and the mean metrics for each boundary loss weight over the whole 2D dataset in Table 12. In these experiments, we use eikonal loss weight 0.1, as provided in the PHASE paper. We also show one example from ShapeNet in Fig. 22, where we use eikonal loss weight 1.0. From our theory and the examples above, one can clearly see that using a small boundary loss weight is not the best strategy.

Second, we design our network to output the signed distance u directly instead of the heat h, whereas PHASE's model would directly output the occupancy o and convert to signed distance. We show that this can lead to extremely numerically unstable results. Consider the case where the occupancy function o outputs 1 or -1, then the log transform (Eq. (41)) would simply output infinity or negative infinity. The infinities can be avoided by clamping the occupancy, but how much should we clamp?

For their proposed setting $\epsilon = 0.01$, only to get $s = 2$, we will need to set $o = 1 - e^{-20} \approx 0.9999999979$, which already is beyond what a 32-bit floating point number can reliably represent. Rescaling the scene is unfortunately not going to help, since the parameter ϵ is scene dependent and needs to be scaled accordingly.

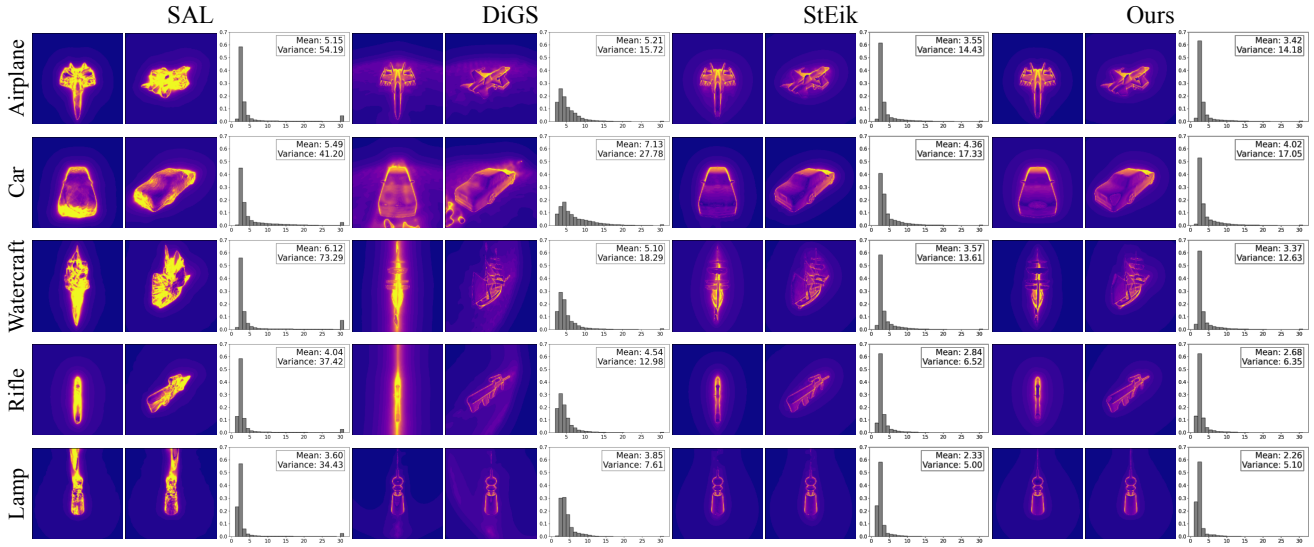
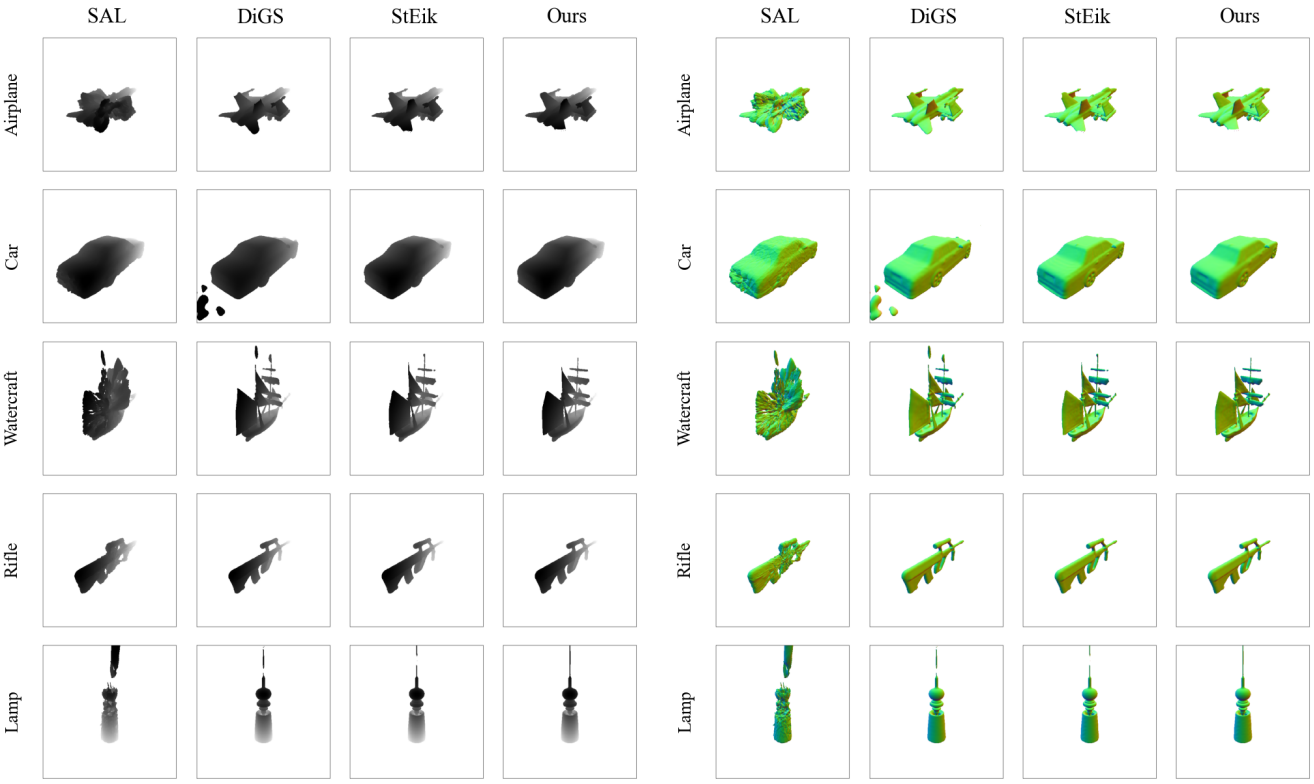


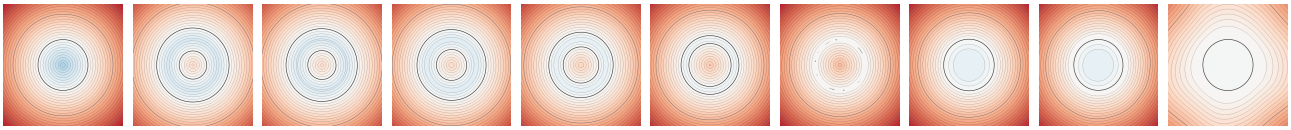
Figure 20. Visualization of iteration counts for each pixel and their corresponding histograms. The iteration count images are taken from two arbitrary poses, while the histograms gather outcomes across all ten rendered poses. Brighter pixels indicate a higher number of queries required for convergence before termination, while darker pixels signify fewer queries. The histograms clearly show that when rendering with our output, most pixels require fewer iterations to determine the surface.



(a) Depth maps showcasing the distance from the camera to the surface.

(b) Normal maps illustrating surface orientation at each point.

Figure 21. Rendering results with sphere tracing algorithm.

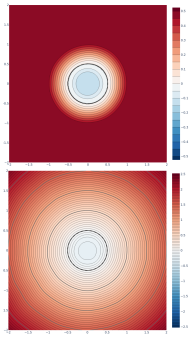


GT	$w_b = 0.1$	$w_b = 0.2$	$w_b = 0.3$	$w_b = 0.5$	$w_b = 1.0$	$w_b = 2.0$	$w_b = 5.0$	$w_b = 10.0$	$w_b = 20.0$
IoU	0.3288	0.3330	0.3242	0.2975	0.2138	0.0022	0.9693	0.9912	0.9917
Chamfer	0.2244	0.2138	0.1955	0.1434	0.0777	0.0175	0.0080	0.0026	0.0025
Hausdorff	0.2315	0.2216	0.1991	0.1455	0.0799	0.1584	0.0150	0.0102	0.0105
RMSE	0.2318	0.2209	0.2085	0.1740	0.1477	0.1476	0.0481	0.0681	0.3004
MAE	0.2237	0.2122	0.1982	0.1552	0.1048	0.0556	0.0336	0.0610	0.2744
SMAPE	0.9159	0.8814	0.8404	0.7059	0.5333	0.3396	0.2028	0.3291	1.0863

Table 11. Comparison of PHASE results on a circle with different w_b values. The color scale is the same as in Fig. 6.

w_b	0.1	0.2	0.3	0.5	1.0	2.0	5.0	10.0	20.0
IoU	0.2026	0.1843	0.2050	0.2089	0.2505	0.2061	0.3899	0.4838	0.4696
Chamfer	0.2663	0.3122	0.2504	0.1785	0.1067	0.0499	0.0793	0.0888	0.1030
Hausdorff	0.5692	0.7001	0.5635	0.4191	0.3446	0.3747	0.4567	0.4383	0.5407
RMSE	0.4644	0.3314	0.2382	0.1606	0.1111	0.0734	0.0567	0.0747	0.1282
MAE	0.4332	0.3114	0.2265	0.1500	0.0918	0.0413	0.0405	0.0623	0.1112
SMAPE	1.2558	1.1504	1.0870	0.9475	0.8107	0.5873	0.7191	0.8877	1.1477

Table 12. Mean metrics of PHASE results with different w_b values on the full 2D dataset.

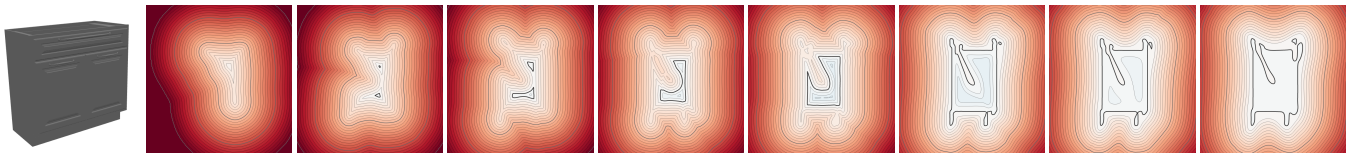


In practice, we verify that when PHASE struggles with queries that are far away from the surfaces, and show visual examples in the inset, where the first figure is PHASE result with $\epsilon = 0.01$ and occupancy clamped at 0.99, and the second figure is our result, which can represent arbitrary signed distance function values.

Moreover, optimizing the occupancy directly instead of the distance leads to another issue when combined with the eikonal loss. When backproping the gradient from s to o, we have:

$$\nabla s = \sqrt{\epsilon}\frac{1}{1-|o|}\nabla o \tag{43}$$

When optimizing the eikonal loss $|\|\nabla s\| - 1|^p$, this $\frac{1}{1-|o|}$ term is multiplied as an coefficient and becomes unstable in the optimization.



GT mesh	$w_b = 0.1$	$w_b = 0.2$	$w_b = 0.3$	$w_b = 0.5$	$w_b = 1.0$	$w_b = 5.0$	$w_b = 10.0$	$w_b = 20.0$
IoU	0.0000	0.0046	0.0387	0.1177	0.3106	0.9742	0.9638	0.9020
Chamfer	0.2056	0.2055	0.1749	0.1441	0.0935	0.0060	0.0065	0.0089
Hausdorff	0.4905	0.4909	0.4395	0.3855	0.3094	0.0518	0.0545	0.0651
RMSE	0.8656	0.8263	0.7944	0.7505	0.6588	0.4578	0.4275	0.4041
MAE	0.8468	0.8108	0.7769	0.7292	0.6339	0.4004	0.3528	0.3182
SMAPE	1.5123	1.5100	1.4956	1.4608	1.3912	1.1019	0.9778	1.0324

Figure 22. Comparison of PHASE results on a 3D cabinet with different w_b values. SDF values are visualized only on a 2D plane. The color scale is the same as in Fig. 6.

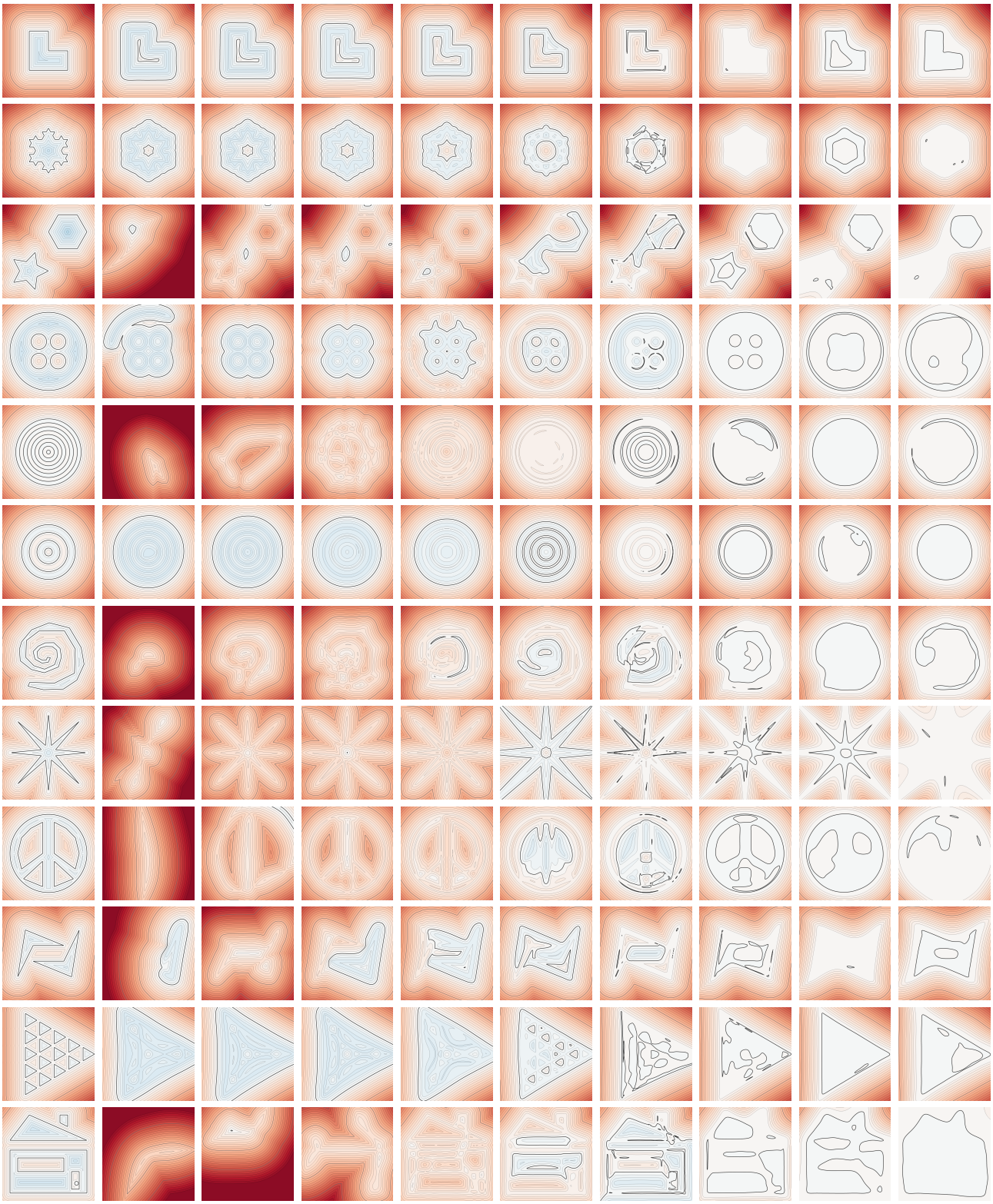


Figure 23. Comparison of PHASE results with different w_b values on the rest of the 2D dataset. The color scale is the same as in Fig. 6.