This MICCAI paper is the Open Access version, provided by the MICCAI Society. It is identical to the accepted version, except for the format and this watermark; the final published version is available on SpringerLink.

Supplementary Material: Speckle-Phase Refinement on an Exact Differentiable Scanner Twin for Inverse OCT Phantom Generation

Daniil Sukhorukov[1,2*], Ekaterina Antipushina[3,4,5*], Yury Koush[3,4], and Ilya Makarov[1,6]

[1] AXXX, Moscow, Russia
[2] ITMO University, Saint Petersburg, Russia
[3] Center for Bio- and Medical Technologies, Moscow, Russia
[4] Skolkovo Institute of Science and Technology, Moscow, Russia
[5] Applied AI Institute, Moscow, Russia
[6] Trusted AI Center, RAS, Moscow, Russia

This supplement expands the two components compressed by the page limit in the main paper: how the conditional U-Net produces the Structured Scatterer Grid (§A) and how it is trained through the differentiable twin (§B). §C details the refinement stage and §D lists the sweeps behind the deployed hyperparameters. Notation follows the main paper.

A From scan to scatterers

A.1 Conditioning input

The network sees two $H \times W$ channels, $\text{cond} = [u_0,\ \tilde{I}]$:

- u_0, the normalized log of the per-scatterer AAI energy. The analytic inversion returns a per-pixel energy map E_{AAI} (Vermeer attenuation calibrated by $E = \mu/c_E$); we divide by K so the K scatterers of a pixel sum to the calibrated μ, clip to $[E_{\min}, E_{\max}] = [0.02, 100]$ Energy-% and map to u_0 by the affine log transform $u = 2\big(\log E - \log E_{\min}\big)/\big(\log E_{\max} - \log E_{\min}\big) - 1$, which sends the admissible energy range onto $[-1, 1]$.
- $\tilde{I} = 2I - 1$, the real scan rescaled to $[-1, 1]$.

Conditioning on u_0 rather than on I alone lets the network predict a *residual* over an initialization that already matches OAC by construction.

A.2 Architecture

F_θ is a U-Net of pre-activation residual blocks (GroupNorm, SiLU, two 3×3 convolutions, 1×1 projection on the skip when widths differ). Table 1 gives the layout: ∼5.3M parameters at base width 48.

The output convolution is initialized to zero in both weight and bias. At step 0 every field is therefore at its neutral value and the assembled phantom equals the AAI warm start exactly, so training begins from a solution that already scores avg MS-SSIM 0.65 instead of from noise.

∗ D. Sukhorukov and E. Antipushina contributed equally to this work.

Table 1. Amortized U-Net F_θ. Downsampling is a stride-2 convolution, upsampling is nearest-neighbour interpolation followed by a 3×3 convolution and concatenation with the matching encoder skip.

Stage	Widths	Resolution
Input convolution	$2 \to 48$	256×512
Encoder (4 residual blocks + 4 downsamplings)	$48, 48, 96, 96, 192$	to 16×32
Bottleneck (2 residual blocks)	192	16×32
Decoder (4 upsamplings + 4 residual blocks, skips)	$192, 96, 96, 48$	to 256×512
Output (GroupNorm, SiLU, 3×3 convolution)	$48 \to 1{+}4K$	256×512

A.3 Decoding the output into fields

The $1{+}4K$ output channels split into one energy channel and K channels for each of δx, δz, δy and the density gate. Table 2 gives the activation and range of each field. Every field is bounded: no output of the network can place a scatterer outside the pixel it was assigned, and no output can drive the energy outside the calibrated envelope by more than a fixed factor.

Table 2. Field decoding. u_0 is the AAI envelope in log units and $\mathcal{E}(\cdot)$ the inverse log transform back to Energy-%.

Field	Channels	Activation	Range
Energy E	1	$\mathcal{E}\big(\mathrm{clip}(u_0 + 0.5\tanh(\cdot), \pm 1.2)\big)$	bounded residual over AAI
δx, δz	$2K$	tanh	$[-1, 1]$ pixel, i.e. $\pm 6\,\mu$m
δy	K	tanh	$[-1, 1]$ beam radius, i.e. $\pm 10\,\mu$m
Gate g	K	$\sigma(\cdot + 3)$	$[0, 1]$, ≈ 0.95 at init

A.4 Assembling the phantom

Base positions are the pixel centres, replicated K times per pixel, giving $M = KHW$ scatterers (262k at $K{=}2$). The fields modulate them as

$$x_j = x_j^0 + \delta x_j\, \Delta, \qquad z_j = \mathrm{clip}\big(z_j^0 + \delta z_j\, \Delta,\ 0,\ Z_{\max}\big), \tag{1}$$
$$y_j = \mathrm{clip}\big(y_j^0 + \delta y_j\, w_0,\ \pm w_0\big), \qquad E_j^{\mathrm{eff}} = \mathrm{clip}\big(E_j\, g_j^2,\ E_{\min},\ E_{\max}\big),$$

with $\Delta = 6\,\mu$m the pixel size and $w_0 = 10\,\mu$m the beam radius. The gate enters squared because the renderer uses $\sqrt{E/100}$ as amplitude, so g scales the amplitude linearly and a gate near zero removes the scatterer from both the coherent sum and the shadowing product. This is the density lever: the scatterer count stays fixed at M (the challenge format wants a fixed-size cloud) while the effective density varies continuously.

B Self-supervised training through the twin

No ground-truth phantoms exist for the real scans, so there is nothing to regress onto. The training signal is the rendering residual: the assembled phantom is rendered by the differentiable twin and compared with the real scan, and the gradient flows back through the renderer into the fields and then into θ. Algorithm 1 states the loop; Table 3 lists the settings.

Algorithm 1 Self-supervised training of F_θ (one epoch)

for each minibatch $\{I_b\}_{b=1}^{4}$ of dev scans **do**
  $E_{\mathrm{AAI}}, u_0 \leftarrow \mathrm{AAI}(I_b)$ ▷ closed form, no gradient
  $\mathrm{cond} \leftarrow [\, u_0,\ 2I_b - 1\,]$
  $\mathrm{out} \leftarrow F_\theta(\mathrm{cond})$
  $E, \delta x, \delta z, \delta y, g \leftarrow \mathrm{decode}(\mathrm{out}, u_0)$ ▷ Table 2
  $P \leftarrow \mathrm{assemble}(E, \delta x, \delta z, \delta y, g)$ ▷ $M = KHW$ scatterers
  $\hat{I} \leftarrow \hat{\Phi}(P)$ ▷ differentiable twin, ∼9 ms batched
  $\mathcal{J} \leftarrow 5\big(1 - \text{MS-SSIM}(\hat{I}, I_b)\big) + 0.3 \sum_{d \in \mathcal{D}} \big(1 - \text{MS-SSIM}(T_d(\hat{I}), T_d(I_b))\big)$
  $\theta \leftarrow \mathrm{AdamW}(\theta, \nabla_\theta \mathcal{J})$
end for

Two points about the objective. The heavy weight sits on the raw B-scan because the physics maps already saturate from the AAI initialization, leaving speckle as the only term with headroom; the 0.3 map term acts as an anchor that keeps OAC, SC and RSC from drifting while the network chases speckle. The perceptual weight is w_{p}=0 here: LPIPS is scored but never back-propagated, since the MS-SSIM terms supply the same gradient direction at lower cost (§D).

The depth ordering inside the shadowing product $\Pi_{a,j}$ depends on z, which the network moves. We re-sort scatterers by depth at every render and treat the sort as a fixed permutation: gradients flow through the gathered values, not through the ordering. The permutation is piecewise constant in z, so this is exact away from ties.

Table 3. Training and refinement settings.

	Amortized training	Speckle-phase refinement
Variables	θ (∼5.3M)	$\delta x, \delta z, \delta y$ ($3M$ values)
Frozen	none	E, g
Optimizer	AdamW, lr 2×10^{-4}, wd 10^{-5}	Adam, lr 0.03
Schedule	cosine to 2%, 400 epochs	cosine, 400 steps
Batch	4 scans	1 scan (per-scan optimization)
Data	80 dev scans	any scan at test time
Cost	offline, once, one GPU	∼30 s, 5.95 GB per scan

C Speckle-phase refinement

Refinement optimizes the same displacement fields the network predicts, starting from its output, with energy and gate frozen. Freezing is what protects the physics maps: $E^{\text{eff}} = Eg^2$ is not a variable of the optimization, so the quantity that sets OAC, SC and RSC keeps the network's prediction exactly.

The displacements stay inside the box of Table 2, which is tight in geometry and loose in phase. A scatterer never leaves its assigned pixel, so the macrostructure the network chose is preserved. Meanwhile the depth phase $e^{-2\mathrm{i}\kappa_n z}$ has period $\lambda_0/2 = 0.65\,\mu$m, so the $\pm 6\,\mu$m range spans ± 9.2 periods and any target phase is reachable inside the constraint. The refinement therefore has full authority over the coherent speckle phase and none over the envelope.

Empirically no map degrades. From the amortized initialization to refinement, held-out Struct goes $0.898 \to 0.955$, OAC $0.977 \to 0.986$, SC $0.993 \to 0.996$ and RSC $0.995 \to 0.997$. Refinement improves every one of the 120 scans, the per-scan standard deviation falls to ± 0.001 and the worst scan scores 0.982. A best-of guard on the exact scanner would make monotonicity formal for one extra render; it never fires, so we leave it out of the deployed generator.

D Sweeps behind the deployed configuration

All sweeps are held-out and scored with the exact scanner.

Table 4. Scatterers per pixel K (avg over the held-out split). K=1 underfits the speckle statistics; $K \geq 3$ costs memory and time without improving, consistent with speckle saturation above two scatterers per resolution cell.

K	1	2	3	4
avg MS-SSIM	0.958	**0.960**	0.959	0.958
Struct	0.877	**0.881**	0.878	0.874

Table 5. Weight of the map anchor. The two points are a Pareto pair: 0.6 buys LPIPS at the cost of MS-SSIM. We deploy 0.3 because the challenge ranks by MS-SSIM.

Anchor weight	0.3	0.6
avg MS-SSIM	**0.9656**	0.9642
LPIPS	0.027	**0.0225**

The perceptual weight took the values $w_{\text{p}} \in \{0, 0.3, 0.5, 1.0\}$ across development runs. Every nonzero setting adds an AlexNet forward and backward pass

per map per step, and none of them beat w_{p}=0 on the exact scanner, so the deployed configuration back-propagates MS-SSIM only and keeps LPIPS as a reported metric.