Supplementary Material for "Soft Shadow Diffusion (SSD): Physics-inspired Learning for 3D Computational Periscopy"

Fadlullah Raji and John Murray Bruce

University of South Florida.
4202 E. Fowler Avenue, Tampa, FL, USA 33620.
fraji@usf.edu murraybruce@usf.edu

This supplemental document to our manuscript provides additional information on the complexity of the purely physics-based approach (Section 1) and its robustness to unknown ambient illumination (Section 2). Additional details on implementing and training our physics-inspired soft shadow diffusion model are provided in Section 3. Finally, Section 4 contains various additional real and synthetic experimental results. It explores the stability of the SSD model to noise and background illumination and also shows additional 3D reconstructions.

1 Physics-based 3D Computational Pericopy: Towards higher resolution voxelizations

Figure 5(b) in the main paper shows a reconstruction from real experimental data using a $10 \times 5 \times 10$ voxels discretization of the *occluding region*. (Note that the portion of the hidden scene which we refer to as the *occluding region* is illustrated in Figure 2 of the main manuscript.)

1.1 Complexity Analysis of Physics-based Reconstruction

The $10 \times 5 \times 10$ pinspecks voxelization is relatively coarse to be able to describe a large variety of 3D objects. Increasing the number of pinspecks comes with increased computational complexity and space requirements. Figure 5(a) in the main manuscript is an example of increasing the grid size which we reconstructed using $16 \times 5 \times 16$ discretization of the occluding region. Given a camera photograph of 128×128, and the non-occluding scene to be estimated at 32×32 resolution, the memory requirement will be $16384 \times 1024 \times 1280 = 80$GB when using Float32 precision. Since our algorithm requires the computation of gradient and updates of the variables to be estimated, this memory requirement doubles. Hence, to achieve the simulated reconstruction in Figure 5(a) of the main manuscript, we used 3 A100 GPU with 80GB each, and distribute the forward model across these GPUs for computation. A sparse representation of this forward model is presented next.

1.2 Sparse Matrix Representation

The contribution (shadow) of a hidden scene pinspeck due to a point light source in the hidden scene may be interpreted using a sparse structure for $\mathbf{V}_k$. Here, the contribution is zero at a few locations and one everywhere else, as shown in Figure 1. More precisely, the one's complement of the contribution is sparse. Thus, to achieve higher resolution, the discrete forward model in main manuscript equation (10) can be evaluated without saving the large matrix $\mathbf{V}_k$ for each pinspeck location indexed by k. Instead, a sparse data structure may be utilized to improve efficiency.

Results of adopting a sparse representation in evaluating the pinspeck forward model at higher reconstruction resolutions are shown in Figure 1. Although we are able to reconstruct correctly in simulation, accurate real-world reconstruction is still challenging. This is, in part, due to the pinspeck approximation. Notwithstanding this shortcoming, this physics-based formulation provides a flexible framework that enables our physics-inspired neural network approach. Our SSD model can handle higher discretizations even for real experimental datasets, as shown in results presented in the main manuscript (Figures 5) and in Supplementary Figure 12 (shown here).

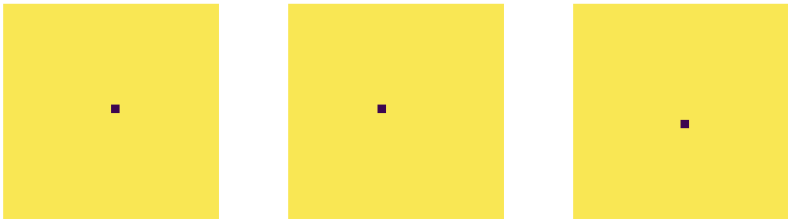


Fig. 1: Each panel shows the variation of the visibility function over the visible wall/camera FOV for some patch in the hidden-scene light-emitting 2D plane and a hidden-scene pinspeck voxel. The visibility function is binary-valued, with one (yellow) indicating that the patch in the hidden-scene light-emitting 2D plane is unoccluded from those patches on the visible wall, and zero (dark blue) indicating occlusion by the pinspeck voxel.

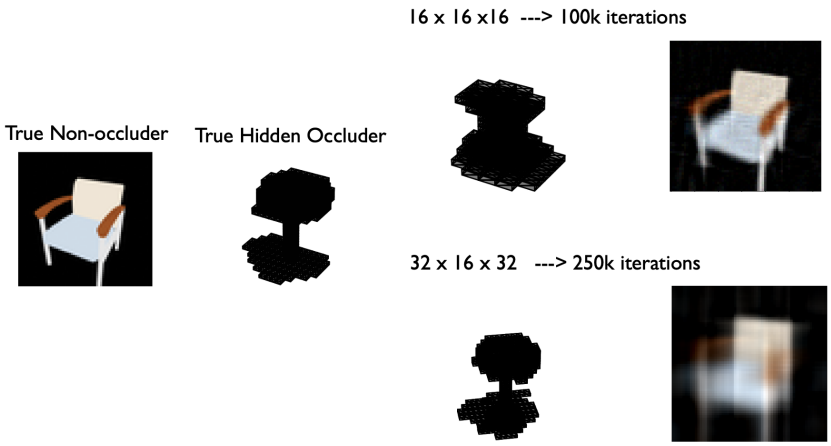


Fig. 2: Increased resolution reconstructions in simulations using sparse data structures.

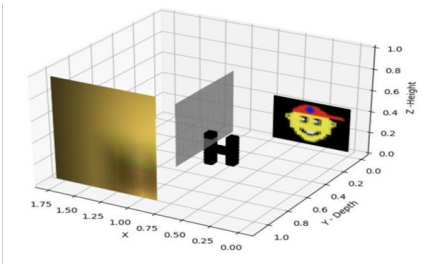
2 Gradient-based inversion: Neglecting background vs Modeling background contributions

We compare the gradient-based inversion via alternating minimization, and a variant that incorporates optimization of the background contribution in Figures 3 and 4. It shows that the solution without learning the noise, and background contribution is blurry. Optimizing the background noise is more efficient, and produces sharp image of the non-occluding structure.

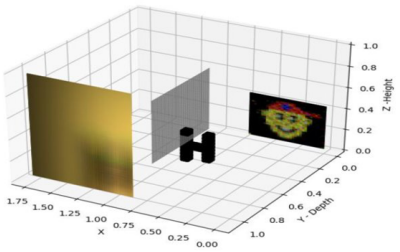
Algorithm 1 Alternating Minimization Method with Background Neglected

Require: $\mathbf{A}, [\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, ..., \mathbf{V}_K], \mathbf{y}, \texttt{numIter}$

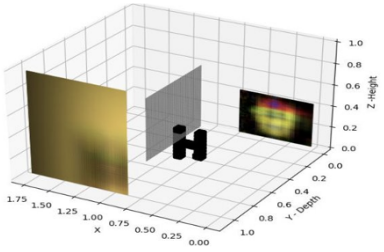
Initialize $\mathbf{z}_0$ randomly, and $\lambda_0 = 1$
Initialize step sizes η_z, and η_λ
for $i = 1$ to `numIter` **do**
  $\mathbf{A}_v = 1 - \frac{1}{K}\sum_{k=1}^{K} \mathbf{V}_k \cdot \sigma(\mathbf{z}_{i_k})$
  $\mathbf{f}^* = (\mathbf{A}_v{}^{\mathsf{T}}\mathbf{A}_v + \lambda_{i-1} I)^{-1}\mathbf{A}_v{}^{\mathsf{T}}\mathbf{y}$
  $\mathbf{f}_i = \max(\mathbf{f}^*, 0)$
  $\mathcal{L}(\mathbf{y}, \mathbf{A}_v\mathbf{f}_i) = \frac{1}{M}\sum_{j=1}^{M} \|\mathbf{y}_j - (\mathbf{A}_v\mathbf{f}_i)_j\|_2^2$
  $\mathbf{z}_i \leftarrow \mathbf{z}_{i-1} - \eta_z \frac{\partial \mathcal{L}(\mathbf{y}, \mathbf{A}_v\mathbf{f}_i)}{\partial \mathbf{z}_{i-1}}$
  $\lambda_i \leftarrow \lambda_{i-1} - \eta_\lambda \frac{\partial \mathcal{L}(\mathbf{y}, \mathbf{A}_v\mathbf{f}_i)}{\partial \lambda_{i-1}}$
end for
Return $\widehat{\mathbf{f}} = \mathbf{f}_i, \widehat{\boldsymbol{\alpha}} = \sigma(\mathbf{z}_i) > 0.5$



(a) Depiction of ground truth scene.



(b) Jointly estimating background contributions.



(c) Without estimating background.

Fig. 3: Reconstructions from gradient-based inversion via alternating minimization. Panels (b) and (c) show the reconstructions of the 3D occluding structures and of the 2D light-emitting scene, with (b) and without (c) incorporating background modeling and estimation.

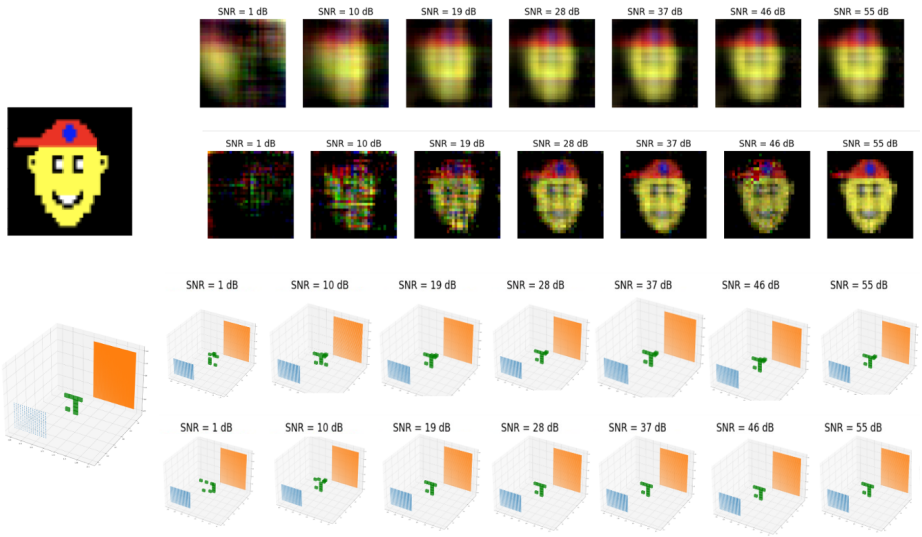


Fig. 4: Comparing the two variants of the gradient-based hidden scene reconstruction algorithm. Row 1: 2D reconstruction of the light-emitting hidden scene component when background contribution **b** is neglected. Row 2: 2D reconstruction of the light-emitting hidden scene component when background contribution **b** is modeled and estimated. Row 3: 3D reconstruction of the light-occluding hidden scene component when background contribution **b** is neglected. Row 4: 3D reconstruction of the light-occluding hidden scene component when background contribution **b** is modeled and estimated.

3 Soft Shadow Diffusion Model

Denote the pinspeck point cloud by $\{\mathbf{u}_k\}_{k=1}^K = \hat{\mathbf{u}}$. The joint distribution $p_\theta(\hat{\mathbf{u}}^{(0:T)})$ is referred to as the reverse process. This process is characterized as a Markov chain with learned Gaussian transitions initiating from $p(\hat{\mathbf{u}}^{(T)}) = \mathcal{N}(\hat{\mathbf{u}}^{(T)}; \mathbf{0}, \mathbf{I})$:

$$p_\theta(\hat{\mathbf{u}}^{(0:T)}) := p(\hat{\mathbf{u}}^{(T)}) \prod_{t=1}^{T} p_\theta(\hat{\mathbf{u}}^{(t-1)}|\hat{\mathbf{u}}^{(t)}), \tag{1}$$

$$p_\theta(\hat{\mathbf{u}}^{(t-1)}|\hat{\mathbf{u}}^{(t)}) := \mathcal{N}(\hat{\mathbf{u}}^{(t-1)}; \mu_\theta(\hat{\mathbf{u}}^{(t)}, t), \sigma^2\mathbf{I}). \tag{2}$$

The approximate posterior $q(\hat{\mathbf{u}}^{1:T}|\hat{\mathbf{u}}^0)$, termed the forward process is defined as a Markov chain that incrementally incorporates Gaussian noise into the data following a predetermined variance schedule $\beta_1, \ldots, \beta_T$:

$$q(\hat{\mathbf{u}}^{(1:T)}|\hat{\mathbf{u}}^0) := \prod_{t=1}^{T} q(\hat{\mathbf{u}}^{(t)}|\hat{\mathbf{u}}^{(t-1)}), \tag{3}$$

$$q(\hat{\mathbf{u}}^{(t)}|\hat{\mathbf{u}}^{(t-1)}) := \mathcal{N}(\hat{\mathbf{u}}^{(t)}); \sqrt{1-\beta_t}\hat{\mathbf{u}}^{(t-1)}, \beta_t\mathbf{I}) \tag{4}$$

Let $\hat{\mathbf{u}} = \{\mathbf{u}_k\}_{k=1}^K$, then the training objective is given as:

$$L(\theta, \phi) = \mathbb{E}_q \Bigg[\sum_{t=2}^{T} \sum_{k=1}^{K} \underbrace{\mathrm{D_{KL}}\left(q(\mathbf{u}_k^{(t-1)}|\mathbf{u}_k^{(t)}, \mathbf{u}_k^{(0)}) \,\middle\|\, p_\theta(\mathbf{u}_k^{(t-1)}|\mathbf{u}_k^{(t)}, \hat{\mathbf{y}}) \right)}_{L_k^{(t-1)}} - \sum_{k=1}^{K} \underbrace{\log p_\theta(\mathbf{u}_k^{(0)}|\mathbf{u}_k^{(1)}, \hat{\mathbf{y}})}_{L_k^{(0)}} + \underbrace{\mathrm{D_{KL}}\left(q_\phi(\hat{\mathbf{y}}|\mathbf{u}^{(0)}) \,\middle\|\, p(\hat{\mathbf{y}}) \right)}_{L_{\hat{\mathbf{y}}}} \Bigg]. \tag{5}$$

3.1 Simplified Training Algorithm

We adapt the simplified algorithm presented in [1] to train our model. To evaluate $L_k^{(t-1)}$, we need to sample $\mathbf{u}_k^{(t)}$ from $q(\hat{\mathbf{u}}^{(t)}|\hat{\mathbf{u}}^{(0)})$. In principle, it can be done by sampling iteratively through the Markov chain. However, [1] showed that $q(\hat{\mathbf{u}}^{(t)}|\hat{\mathbf{u}}^{(0)})$ is Gaussian, thus allowing us to sample $\hat{\mathbf{u}}^{(t)}$ efficiently without iterative sampling. To see this, note that:

$$q(\hat{\mathbf{u}}^{(t)}|\hat{\mathbf{u}}^{(0)}) = \mathcal{N}(\hat{\mathbf{u}}^{(t)}|\sqrt{\bar{\alpha}_t}\hat{\mathbf{u}}^{(0)}, (1-\bar{\alpha}_t)I). \tag{6}$$

Since both $q(\mathbf{u}_k^{(t-1)}|\mathbf{u}_k^{(t)}, \mathbf{u}_k^{(0)})$ and $p_\theta(\mathbf{u}_k^{(t-1)}|\mathbf{u}_k^{(t)}, \hat{\mathbf{y}})$ are Gaussians, the term $L_k^{(t-1)}$ can be expanded as:

$$L_k^{(t-1)} = \mathbb{E}_{\mathbf{u}_k^{(0)}, \mathbf{u}_k^{(t)}, \hat{\mathbf{y}}} \left[\frac{1}{2\beta_t} \left\| \frac{\sqrt{\bar{\alpha}_{t-1}\beta_t}}{1-\bar{\alpha}_t}\mathbf{u}_k^{(0)} + \frac{\sqrt{\alpha_t(1-\bar{\alpha}_{t-1})}}{1-\bar{\alpha}_t}\mathbf{u}_k^{(t)} - \mu_\theta(\mathbf{u}_k^{(t)}, t, \hat{\mathbf{y}}) \right\|^2 \right] + C. \tag{7}$$

Using the Gaussian above, $\mathbf{u}_k^{(t)} = \sqrt{\bar{\alpha}_t}\mathbf{u}_k^{(0)} + \sqrt{1-\bar{\alpha}_t}\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$:

$$L_k^{(t-1)} = \mathbb{E}_{\mathbf{u}_k^{(0)},\epsilon,\hat{\mathbf{y}}}\left[\frac{1}{2\beta_t}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{u}_k^{(t)} - \beta_t\sqrt{1-\bar{\alpha}_t}\epsilon\right) - \mu_\theta(\mathbf{u}_k^{(t)}, t, \hat{\mathbf{y}})\right\|^2\right] + C. \quad (14)$$

(8)

Shown above $\mu_\theta(\mathbf{u}_k^{(t)}, t, \hat{\mathbf{y}})$ must predict $\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{u}_k^{(t)} - \beta_t\sqrt{1-\bar{\alpha}_t}\epsilon\right)$ given $\mathbf{u}_k^{(t)}$. Thus, $\mu_\theta(\mathbf{u}_k^{(t)}, t, \hat{\mathbf{y}}))$ can be parameterized as:

$$\mu_\theta(\mathbf{u}_k^{(t)}, t, \hat{\mathbf{y}}) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{u}_k^{(t)} - \beta_t\sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{u}_k^{(t)}, t, \hat{\mathbf{y}})\right), \quad (9)$$

where $\epsilon_\theta(\mathbf{u}_k^{(t)}, t, \hat{\mathbf{y}})$ is the physics-inspired neural network intended to predict ϵ from $\mathbf{u}_k^{(t)}$. Finally, $L_k^{(t-1)}$ can be simplified as

$$L_k^{(t-1)} = \mathbb{E}_{\mathbf{u}_k^{(0)},\epsilon,\hat{\mathbf{y}}}\left[\frac{\beta_t^2}{2\beta_t\alpha_t(1-\bar{\alpha}_t)}\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{u}_k^{(0)} + \sqrt{1-\bar{\alpha}_t}\epsilon, t, \hat{\mathbf{y}})\right\|^2\right] + C. \quad (14)$$

(10)

To minimize $L_i^{(t-1)}$, we can only minimize $\mathbb{E}[\|\epsilon - \epsilon_\theta\|^2]$ because the coefficient $\frac{\beta_t^2}{2\beta_t\alpha_t(1-\bar{\alpha}_t)}$ is constant.

The simplified algorithm [1] proposed selecting a random term from the set $\left\{\sum_{k=1}^K L_k^{(t-1)}\right\}_{t=1}^T$ for optimization in each step of the training process.

Then the training algorithm is presented in Algorithm 2.

Algorithm 2 SSD Training algorithm

1: **repeat**
2: Sample the occluding components $\{\mathbf{u}_k^{(0)}\}_{k=1}^K \sim p_{\text{data}}(\mathbf{u}^{(0)})$
3: Sample the non-occluding components $\mathbf{f} \in \mathbb{R}^N \sim p_{\text{data}}(\mathbf{f})$
4: Generate the measurement $\mathbf{y} = \mathbf{A}\left(1 - \sum_{k=1}^K \mathbf{u}_k^{(0)}\right)\mathbf{f}$
5: Sample $\hat{\mathbf{y}} \sim q_\phi(\mathbf{y}|\hat{\mathbf{u}}^{(0)})$
6: Sample $t \sim \text{Uniform}(\{1, \ldots, T\})$
7: Sample $\epsilon \sim \mathcal{N}(0, I)$
8: Compute $\nabla\left[\sum_{k=1}^K \left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{u}_k^{(0)} + \sqrt{1-\bar{\alpha}_t}\epsilon, t, \hat{\mathbf{y}}\right)\right\|^2\right]$;
9: Perform gradient descent.
10: **until** converged

3.2 Inference Algorithm

Given $\mathbf{y}$ inference is performed using SSD according to Algorithm 3.

Algorithm 3 SSD Inference algorithm

Given the soft shadow photograph $\mathbf{y}$
Encode the photograph: $\hat{\mathbf{y}} \leftarrow F_\alpha(\mathbf{y})$
Sample noise from Gaussian distribution $\{\mathbf{u}_k^{(T)}\}_{k=1}^K \leftarrow \hat{\mathbf{u}}^{(T)} \sim \mathcal{N}(0, I)$
for $t = T, \ldots, 1$ **do**
    Denoise
    Sample $\{\mathbf{u}_k^{(t-1)}\}_{k=1}^K \sim p_\theta(\{\mathbf{u}_k^{(t-1)}\}_{k=1}^K | \{\mathbf{u}_k^{(t)}\}_{k=1}^K, \hat{\mathbf{y}})$
end for
return $\{\mathbf{u}_k^{(0)}\}_{k=1}^K$

3.3 Implementation Details

Soft Shadow Image Encoder. The architecture of our encoder follows that of crossformer [2] with a number of transformer blocks (2, 4, 8, 2), shown in Table 1.

Dataset Preparation. We simulated 262,000 measurements and sampled the corresponding pointclouds. From them, we randomly selected 3,000 examples as the test set, the remaining 259,000 are used as the training set.

Diffusion Process. The number of steps in the diffusion process is $T = 256$. We set the variance schedules β_t following a cosine schedule.

The training parameters are shown in Table 2.

U-Net-Based Diffusion Model. The model architecture is based on the Unet design, as detailed in Figure 5. This architecture is particularly suited for handling point cloud data due to its ability to efficiently process spatial hierarchies and feature representations. In our implementation, the model is trained on point cloud tensors with a shape of (2048, 3), representing a set of 2048 points in a three-dimensional space.

To enhance the model's capability in reconstructing detailed point clouds, we employ a sampling strategy that generates a denser point cloud. Specifically, during the reconstruction phase, we sample a higher number of points, aiming for 15,000 points. This approach allows for generating a more detailed and dense representation of the reconstructed occluding structure, capturing finer nuances and structures that are crucial for high-fidelity mesh representation.

Table 1: CrossFormer Model for Soft Shadow Image Encoding (CrossFormer-T)

Stages	Output Size	Layer Name	CrossFormer-T
Stage-1	32×32	Cross Embed.	Kernel size: $\begin{bmatrix} 4 \times 4 \\ 8 \times 8 \\ 16 \times 16 \\ 32 \times 32 \\ Stride = 4 \end{bmatrix}$
		SDA/LDA/MLP	$\begin{bmatrix} D1 = 64 \\ H1 = 2 \\ G1 = 7 \\ I1 = 8 \end{bmatrix} \times 2$
Stage-2	16×16	Cross Embed.	Kernel size: 4×4, Stride=4 (S1 = 56)
		SDA/LDA/MLP	$\begin{bmatrix} D2 = 128 \\ H2 = 4 \\ G2 = 7 \\ I2 = 4 \end{bmatrix} \times 4$
Stage-3	8×8	Cross Embed.	Kernel size: 2×2, Stride=2 (S2 = 28)
		SDA/LDA/MLP	$\begin{bmatrix} D3 = 256 \\ H3 = 8 \\ G3 = 7 \\ I3 = 2 \end{bmatrix} \times 8$
Stage-4	4×4	Cross Embed.	Kernel size: 2×2, Stride=2 (S3 = 14)
		SDA/LDA/MLP	$\begin{bmatrix} D4 = 512 \\ H4 = 16 \\ G4 = 7 \\ I4 = 1 \end{bmatrix} \times 2$
		Cross Embed.	Kernel size: 2×2, Stride=2 (S4 = 7)
Head	1×1	Avg Pooling Kernel size: 4×4	
		Linear Latent: 512	

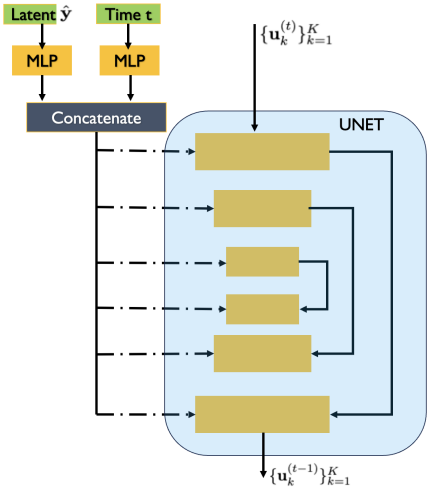


Fig. 5: SSD UNet Module

Table 2: SSD Training Details

Parameter	Value
Base channels	128
Optimizer	Adam
Channel multipliers	1, 2, 4,8
Learning rate	1×10^{-4}
Blocks per resolution	2
Batch size	192
Attention resolutions	256, 128
EMA	0.9999
Attention Number of heads	8
Dropout	0.0
Conditioning embedding dimension	512
Training hardware	3 × A100(80G)
Conditioning embedding MLP layers	1
Time embedding MLP layers	1
Training Iterations	600000
Diffusion noise schedule	Cosine
Weight decay	0.01
Sampling timesteps	256
Loss	Mean Squared Error (L2)

4 Results

4.1 Background Lights and Distortions

To investigate the influence of background and noise on our method, we incorporated a controlled light source within the experimental setup. The quantitative results are presented in Figure 6(b) of the main paper. Figure 6 presents the qualitative results of this experiment. We also simulated a noise process on the real measurement and the result is presented in Figure 7. Simulated examples are shown in Figures 8 and 9

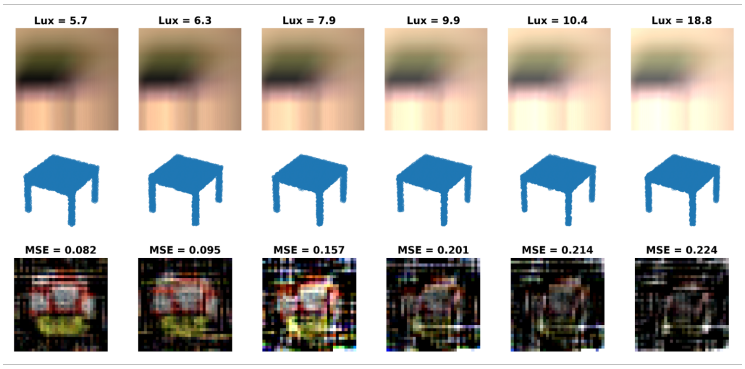


Fig. 6: Experimental Results with Varying Background Lights

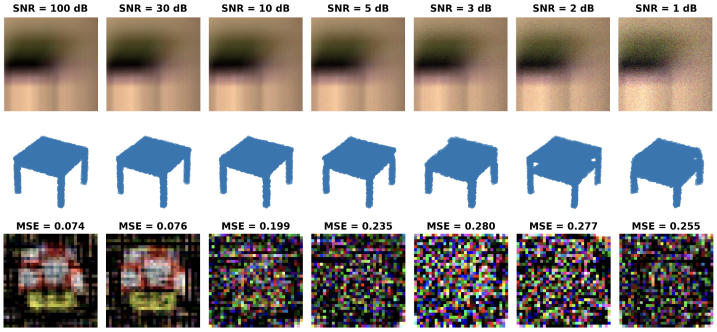


Fig. 7: Experimental Results with Varying Signal-to-Noise Ratio (SNR)

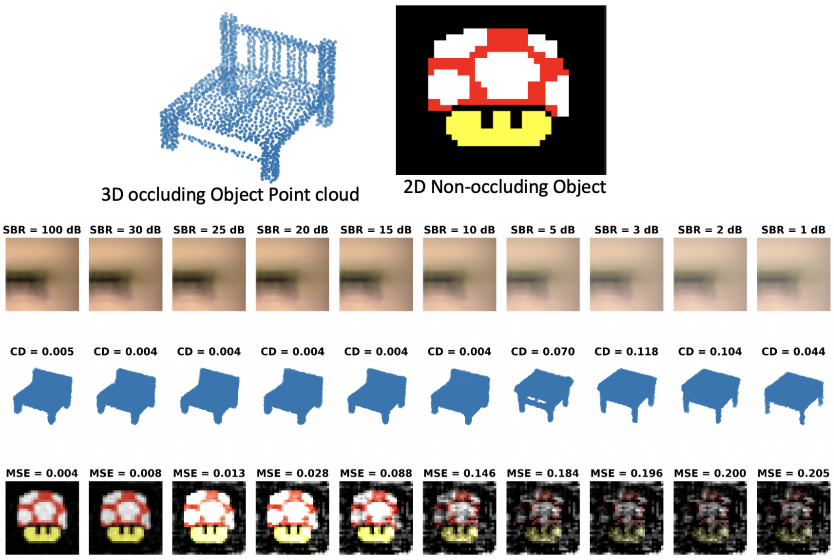


Fig. 8: Simulated Results with Varying Signal-to-Background Ratio (SBR)

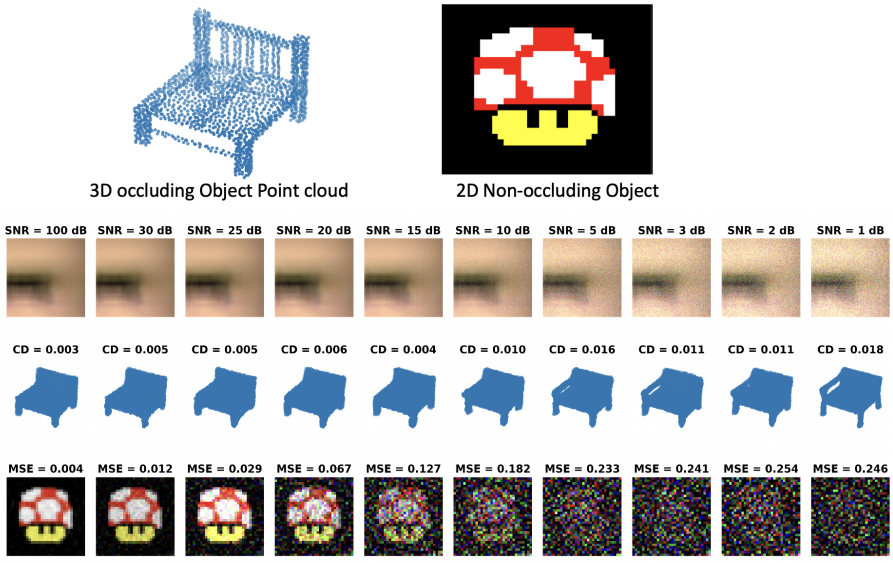


Fig. 9: Simulated Results with Varying Signal-to-Noise Ratio (SNR)

4.2 Shape Reconstructions

Additional results are provided here to demonstrate the performance of the proposed SSD model on a variety of shape classes. We sampled 64 meshes outside our training dataset, and simulated the penumbra photograph using random 2D non-occluding objects. The result of this experiment is shown in Figure 10.

We recall that the model is trained in simulation. Results for simulated experimental data are shown in Figures 11 and 13 to 15, while additional results

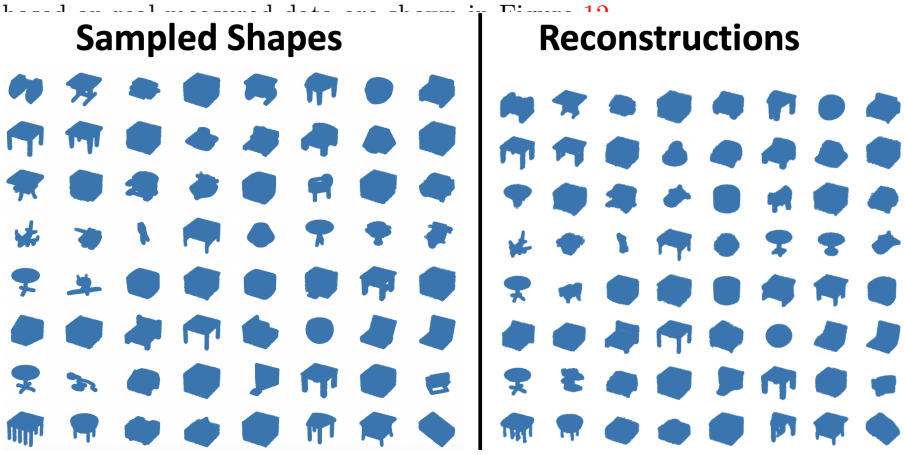


Fig. 10: Reconstructed shapes outside the Training dataset.

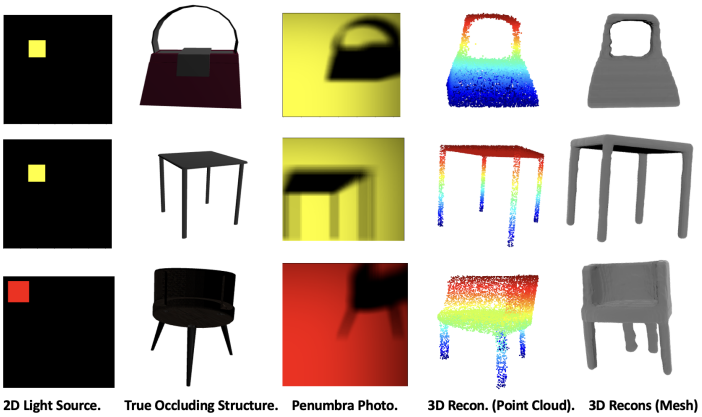


Fig. 11: SSD Reconstruction from Hard Shadows.

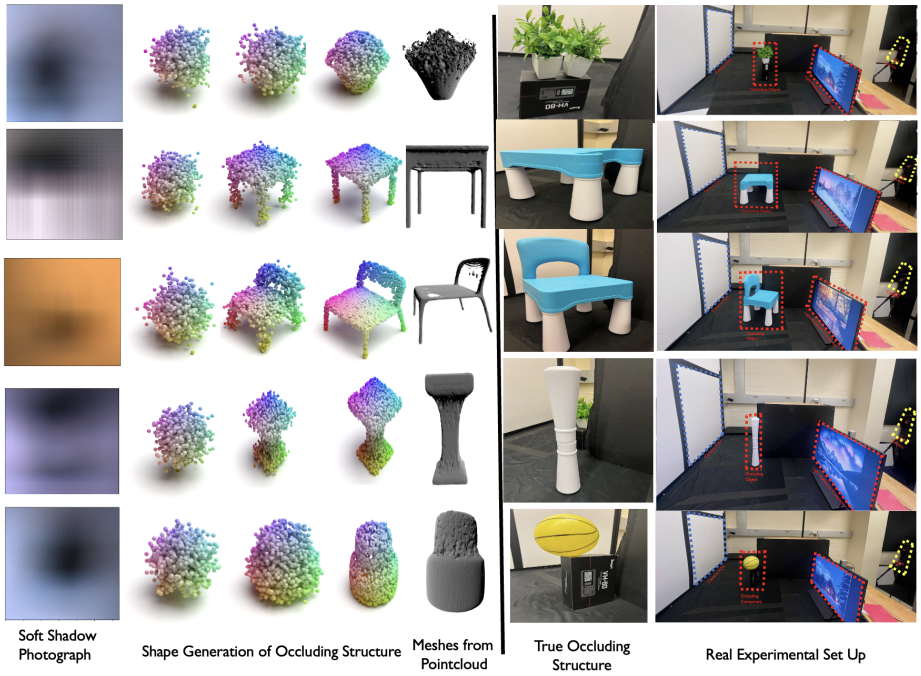


Fig. 12: Results from SSD in Real Experimental Setup.

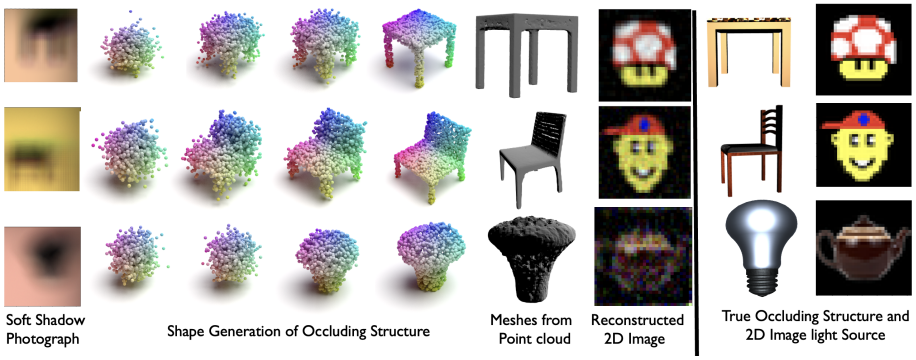


Fig. 13: Results from SSD in Simulation.

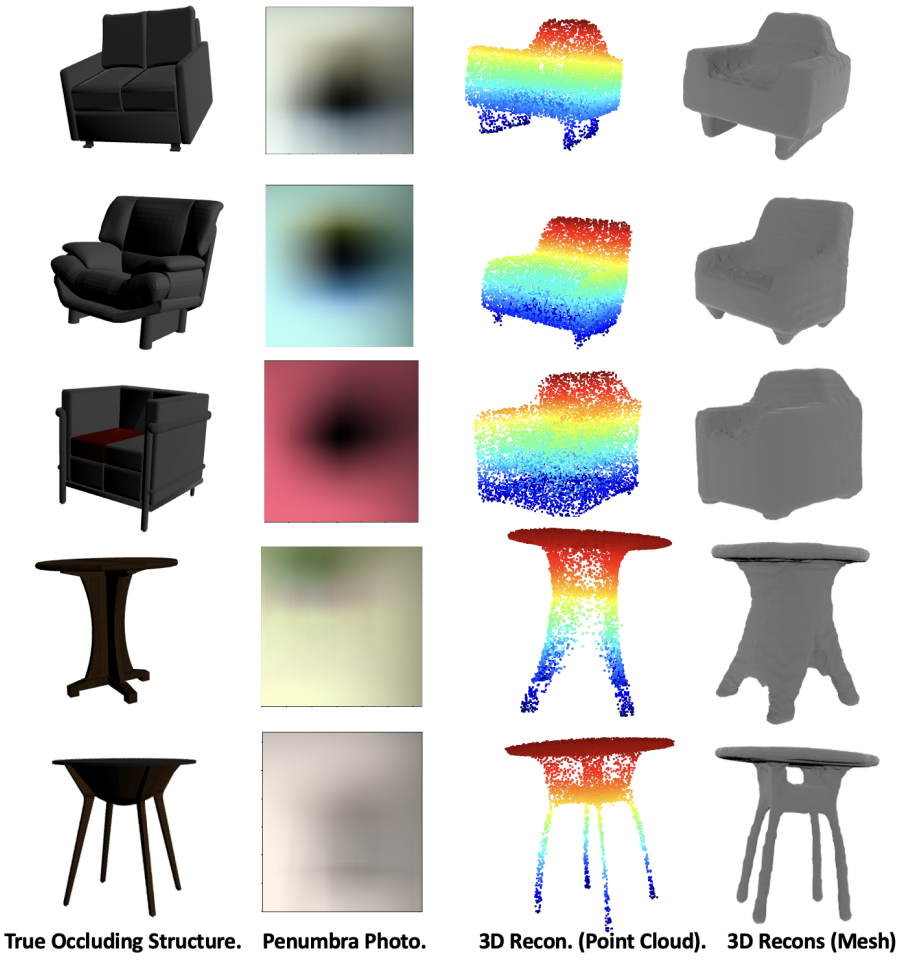


Fig. 14: Results from SSD in Simulation.

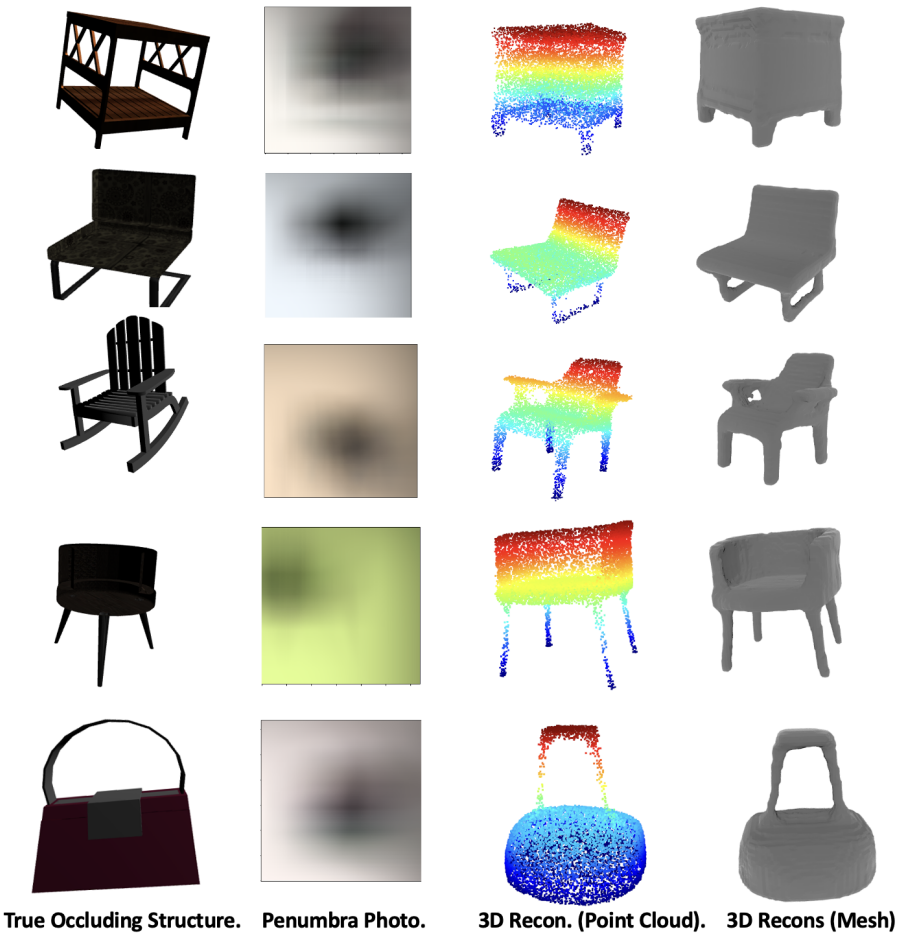


Fig. 15: Results from SSD in Simulation.

References

1. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems (NeurIPS) **33**, 6840–6851 (2020) 7, 8
2. Wang, W., Yao, L., Chen, L., Lin, B., Cai, D., He, X., Liu, W.: Crossformer: A versatile vision transformer hinging on cross-scale attention. In: Proc. Int. Conf. on Learning Representations (ICLR) (2022) 9