Supplementary Material for

MDUNet: Multimodal Decoding UNet for Passive Occluder-Aided Non-line-of-sight 3D Imaging

A. Implementation and Training Details

A.1. Overview

In this supplementary material, we provide additional details about our training procedures, model architectures, and hyperparameter choices for both the SDF autoencoder (used to represent 3D occluders) and MDUNet.

A.2. Hardware and Software Environment

Model is implemented in Python using PyTorch. We train on 3 NVIDIA A100 GPUs with 80 GB of memory each. Ray-mesh intersections and SDF sampling rely on CUDA kernels for speed. When GPU resources are unavailable, CPU-based fallbacks, Open3D used for mesh processing and point-cloud operations. Model has a total of 250M parameters.

A.3. Data Generation and Preprocessing

As presented in Algorithm 1, we create physically accurate training data for passive NLOS reconstruction, by procedurally generating synthetic scenes composed of both occluding and non-occluding elements. For each scene, we randomly sample between 2 and 10 textured 3D objects from the Objaverse dataset to serve as the non-occluding components, and a single untextured mesh from the ShapeNet dataset to act as the occluder. These objects are randomly placed within a constrained hidden region behind the visible measurement wall. Additionally, we include a distant emissive background surface positioned behind the non-occluding scene to simulate realistic ambient illumination conditions.

We render soft shadow photographs by discretizing the measurement wall into an $M \times M$ grid and casting a bundle of $N \times N$ rays from each pixel into the hidden scene. Each ray is traced to determine its first intersection point. If the ray strikes the occluder, no radiance is accumulated due to full visibility occlusion. If it intersects either a non-occluding object or the background emitter, we compute its contribution using a Lambertian model that accounts for surface reflectance, angular falloff, and inverse-square distance attenuation. The total irradiance at each wall pixel is obtained by summing over all unoccluded ray contributions.

To supervise the model's reconstruction outputs, we also generate a canonical 2D projection of the non-occluding scene and extract a normalized point cloud from the occluder mesh. This process enables the creation of diverse, physically grounded examples that support robust learning under complex occlusion and illumination settings.

- **Occluding Scenes.** We use a combination of $\sim$53,000 meshes from ShapeNet and 83,000 filtered from Objaverse to form our occluder dataset. For each shape, we sample $\sim$2048 points on the mesh surface to the ground truth occluder for SDF latent. This point cloud is normalized to lie within the cube $[-0.5, 0.5]^3$.
- **Non-Occluding Scenes.** We use $\sim$83,000 textured meshes from Objaverse for the non-occluding scenes. We project meshes to a 64×64 image by placing a virtual camera at a fixed point on the visible wall. This image serves as the ground truth for the 2D radiosity map.
- **Soft Shadow Photographs.** To simulate each training instance, we place one ShapeNet occluder and one Objaverse mesh in a random arrangement behind the visible wall. We then cast 128×128 rays per pixel to compute the soft shadow irradiance. We add background light levels in simulated datasets and optionally inject additive Gaussian noise to mimic real-world conditions.

A.4. Simulation Parameters

B. SDF Autoencoder Architecture

B.1. Model Components

PointNet-based Triplane Projector. To facilitate accurate reconstruction of occluder geometry, we pretrain a conditional SDF autoencoder utilizing the complete ShapeNet dataset of approximately 53,000 meshes. Each mesh is normalized to reside within a unit cube spanning $[-1, 1]^3$, ensuring consistency across diverse object categories. For every normalized mesh, we sample ground truth SDF values on a uniform $128 \times 128 \times 128$ voxel grid, providing dense supervision for training the decoder. The latent geometric representation of each mesh is encoded via a PointNet-based triplane projector. This encoder consists of three layers with output dimensions of 32, 64, and 128, each followed by ReLU activation functions. Point-wise features are aggregated using a differentiable scatter operation (`torch-scatter`) and projected onto three orthogonal 2D planes aligned with the xy, yz, and zx axes. Each plane has a resolution of 128×128 with 16 feature channels, resulting in a triplane representation $\Phi_{\text{tri}} \in \mathbb{R}^{32\times3\times128\times128}$, which is reshaped into $\mathbb{R}^{96\times128\times128}$ to interface with the downstream decoder. This compact representation effectively encodes 3D geometry while enabling efficient conditioning of the SDF prediction

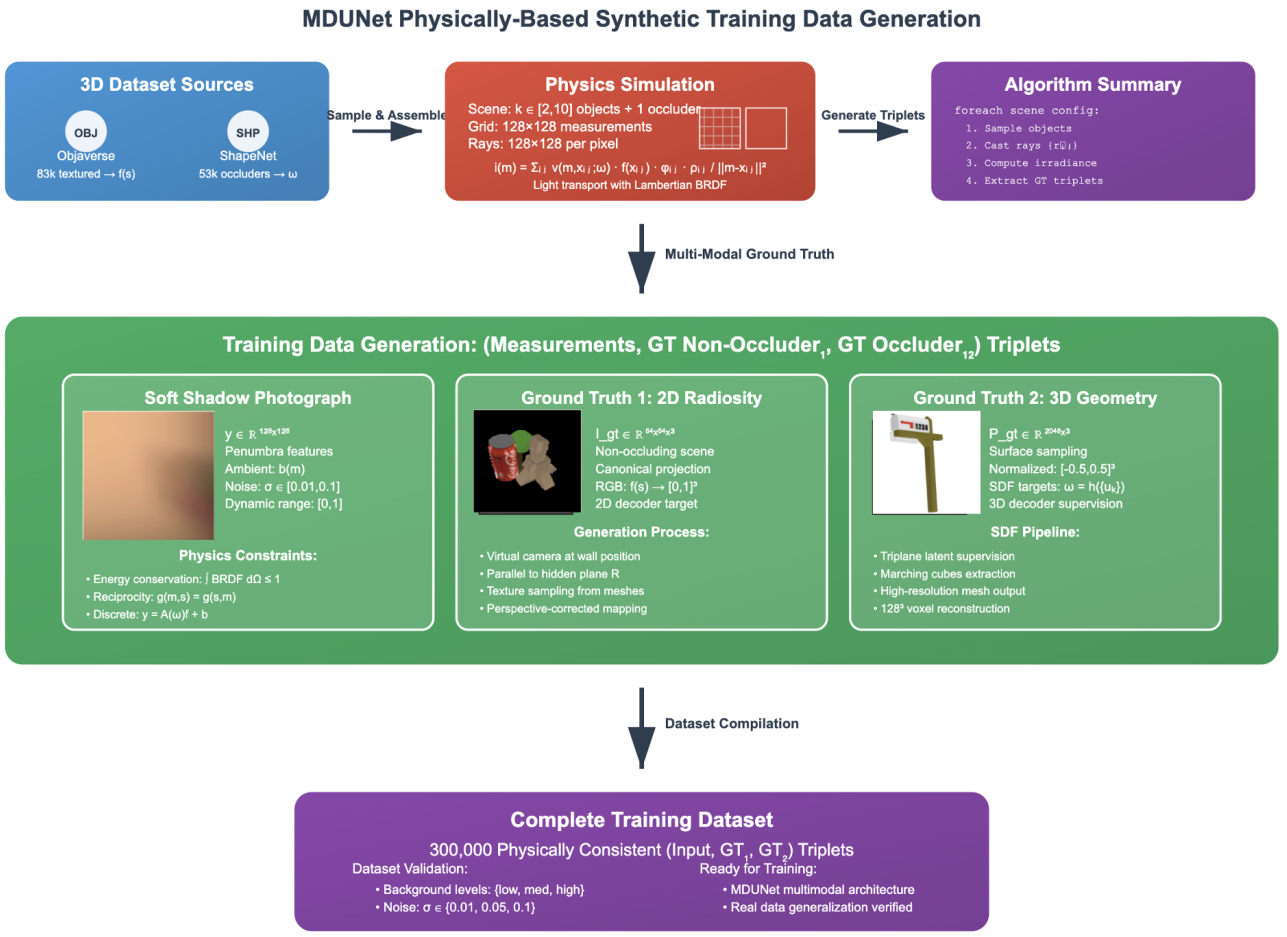

Figure 6. **MDUNet Synthetic Data Generation Pipeline.** Physics-based simulation generates 300,000 training triplets from 3D asset libraries. Ray tracing on 128×128 grids computes light transport to produce soft shadow images, 2D radiosity maps of non-occluding scenes, and 3D occluder point clouds. The pipeline enforces physical constraints including energy conservation and Lambertian BRDF modeling, with validation across multiple noise and illumination conditions for real data generalization.

model on input point clouds.

Encoder and Decoder.

- **Encoder:** A small 2D convolutional network maps Φ_{tri} into a 3D latent vector $z \in \mathbb{R}^3$. Concretely, we use 4 convolution layers with 3×3 kernels, 32–128 channels, and stride 2 for downsampling, ending with a global average pooling and a linear layer to produce z. The shape of z is $16 \times 16 \times 16$ for all experiments.
- **Decoder:** The decoder inverts the above process, expanding z back into triplane features $\widehat{\Phi}_{\text{tri}}$ via transposed convolutions. We mirror the encoder's structure but in reverse. The final output has the same dimension as Φ_{tri}.

MLP for Implicit SDF.

- We query the reconstructed triplane $\widehat{\Phi}_{\text{tri}}$ at a spatial location corresponding to a query points in a discretized $128 \times 128 \times 128$ within $[-0.5, 0.5]^3$, and for each point $\mathbf{q}_p \in \mathbb{R}^3$. we project $\mathbf{q}_p$ onto each of the three planes and bilinearly sample the features.
- A 5-layer MLP (width 128 per layer, ReLU activations) then predicts the SDF value given the decoded triplane feature $f_{\text{SDF}}(\mathbf{q}_p, \widehat{\Phi}_{\text{tri}})$.

B.2. Training SDF Autoencoder

- **Loss Function:**

$$\mathcal{L}_{\text{3D}} = \left\| \text{Recon}(\mathcal{Q}, \mathbf{q}_p) - \text{SDF}(\mathbf{q}_p) \right\|_1 + \lambda \left\| z - \tanh(z) \right\|_1,$$

with $\lambda = 0.1$. for each batch in training, We sample $\sim$48k query points $\mathbf{q}_p$ inside the bounding cube $[-0.5, 0.5]^3$.

- **Optimizer and Schedule:** We use Adam with learning rate 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$. Training runs for 250k

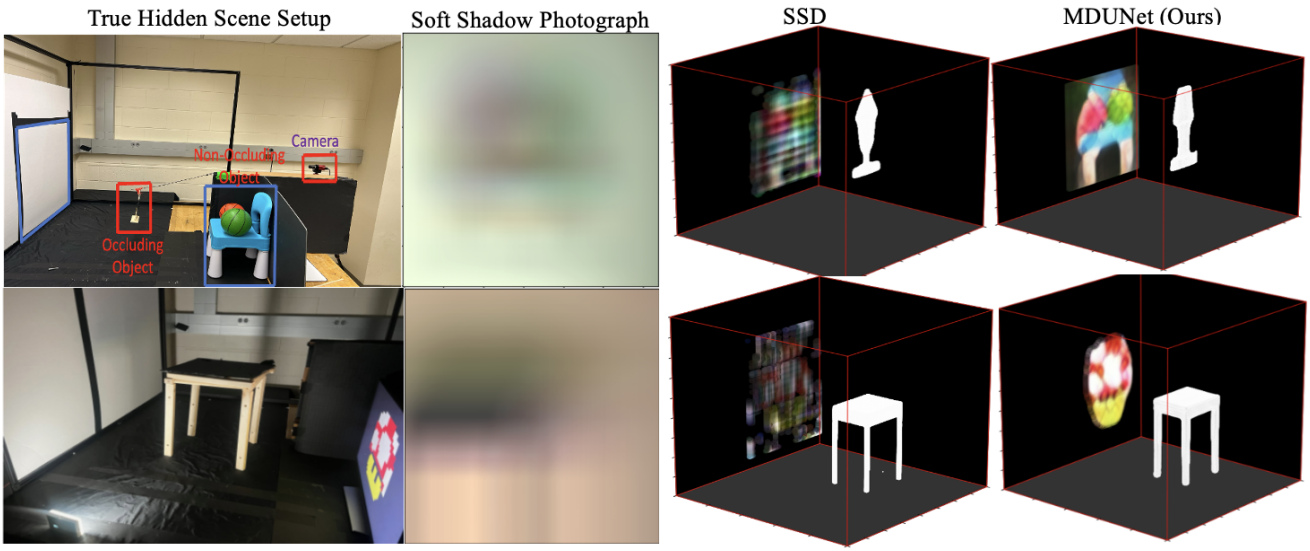

Figure 7. Hidden Scene Reconstructions. A second viewpoints of our 3D reconstructions.

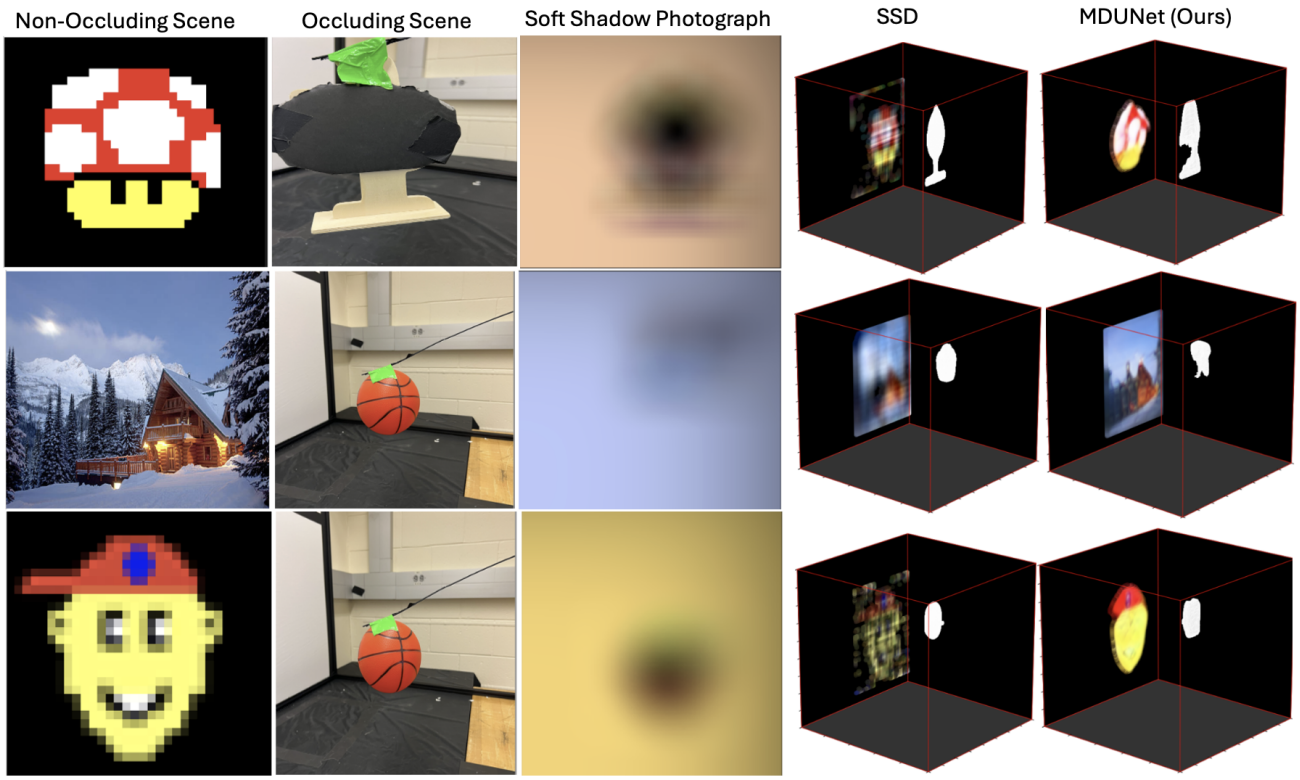

Figure 8. Hidden Scene Reconstructions. A second viewpoints of our 3D reconstructions.

iterations with a batch size of 16 on each GPU and 96 global batch size.

- **Latent Regularization:** The term $\|z - \tanh(z)\|_1$ restricts z to remain within $[-1, 1]$, ensuring stable decoding and avoiding degenerate SDF solutions.

C. MDUNet Architecture

C.1. Shared Encoder

- The first half of the network is a 2D encoder that processes 128×128 soft shadow photographs. We use multiple convolution blocks, self-attention, and leaky ReLU

Algorithm 1 Synthetic Scene Generation

Input: Objaverse dataset $\mathcal{D}_{\text{obj}}$, ShapeNet dataset $\mathcal{D}_{\text{occ}}$, measurement resolution $M \times M$, rays per pixel $N \times N$
Output: Soft shadow image $y \in \mathbb{R}^{M \times M}$, ground truth 2D image I_{gt}, occluder point cloud P_{gt}

Sample k non-occluding objects $S_{\text{nonocc}} \subset \mathcal{D}_{\text{obj}}$
Sample 1 occluding mesh $S_{\text{occ}} \sim \mathcal{D}_{\text{occ}}$
Add a distant, emissive background surface S_{bg} (e.g., a patch behind the non-occluders)
Randomly place all elements in the hidden scene volume

foreach *pixel* $m \in M \times M$ **do**
  Initialize $y[m] \leftarrow 0$
  for $i, j = 1$ **to** $N \times N$ **do**
    Cast ray $r_{i,j}$ from m into hidden region
    $x \leftarrow \text{Trace}(r_{i,j})$
    if x *hits* S_{occ} **then**
      $v \leftarrow 0$
    else if x *hits* S_{nonocc} *or* S_{bg} **then**
      $v \leftarrow 1$
      Retrieve emitted radiance $f(x)$ and surface normals n_x, n_m
      $\varphi \leftarrow \langle \widehat{m - x}, n_m \rangle, \quad \rho \leftarrow \langle \widehat{x - m}, n_x \rangle$
      $d \leftarrow \|m - x\|^2$
      $y[m] \mathrel{+}= v \cdot f(x) \cdot \varphi \cdot \rho / d$

Render I_{gt} from canonical view of S_{nonocc}
Extract and normalize point cloud P_{gt} from S_{occ}
return y, I_{gt}, P_{gt}

activations.

C.2. Two Decoding Paths

Path A (2D Decoder).

- The first decoder reconstructs the 2D radiosity map (64×64) of the non-occluding scene. We apply transposed convolutions (mirroring the encoder) with skip connections at each resolution scale.
- The last layer has a 3-channel output for RGB color. We apply a $\tanh$ activation to keep pixel values within $[-1, 1]$ before rescaling to $[0, 1]$.

Path B (3D Decoder).

- The second decoder outputs a latent code $\hat{z}$ of shape $16 \times 16 \times 16$) for the SDF autoencoder.
- Once $\hat{z}$ is obtained, we pass it into the SDF autoencoder's decoder to generate tri-plane features $\widehat{\Phi}_{\text{tri}}$. Then, for each query point, an MLP produces $f_{\text{SDF}}(\mathbf{q}_p, \widehat{\Phi}_{\text{tri}})$.

D. Additional Results

We further show the impact of visible wall texture on a new dataset in Fig. 11, where MDUNet outperforms SSD under challenging out-of-distribution dataset. We also show real experimental variation of the ambient illumination light in Fig, 12.

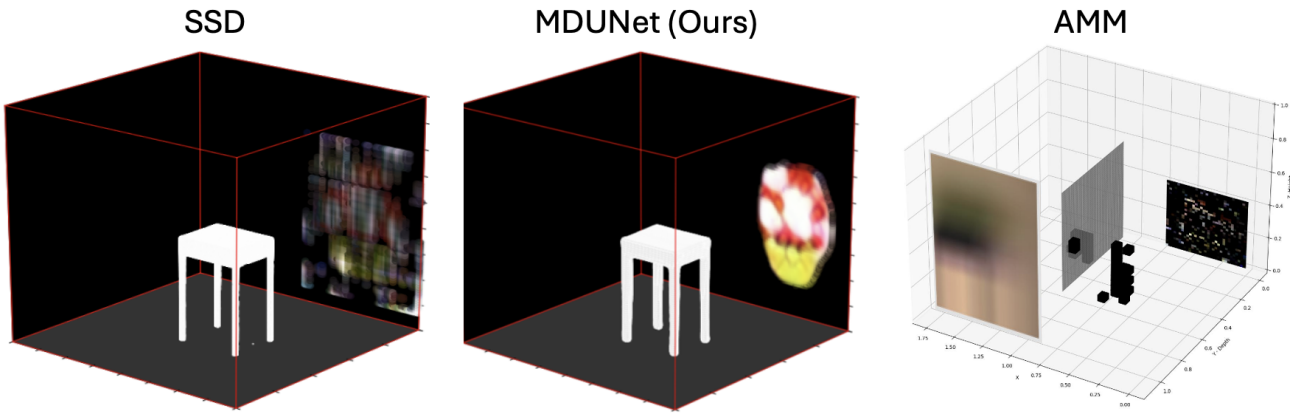

Figure 9. Hybrid Approach ([43] vs Full Learning Based Method (Ours) vs Classical Alternating Minimization Method (AMM)[44, 45]

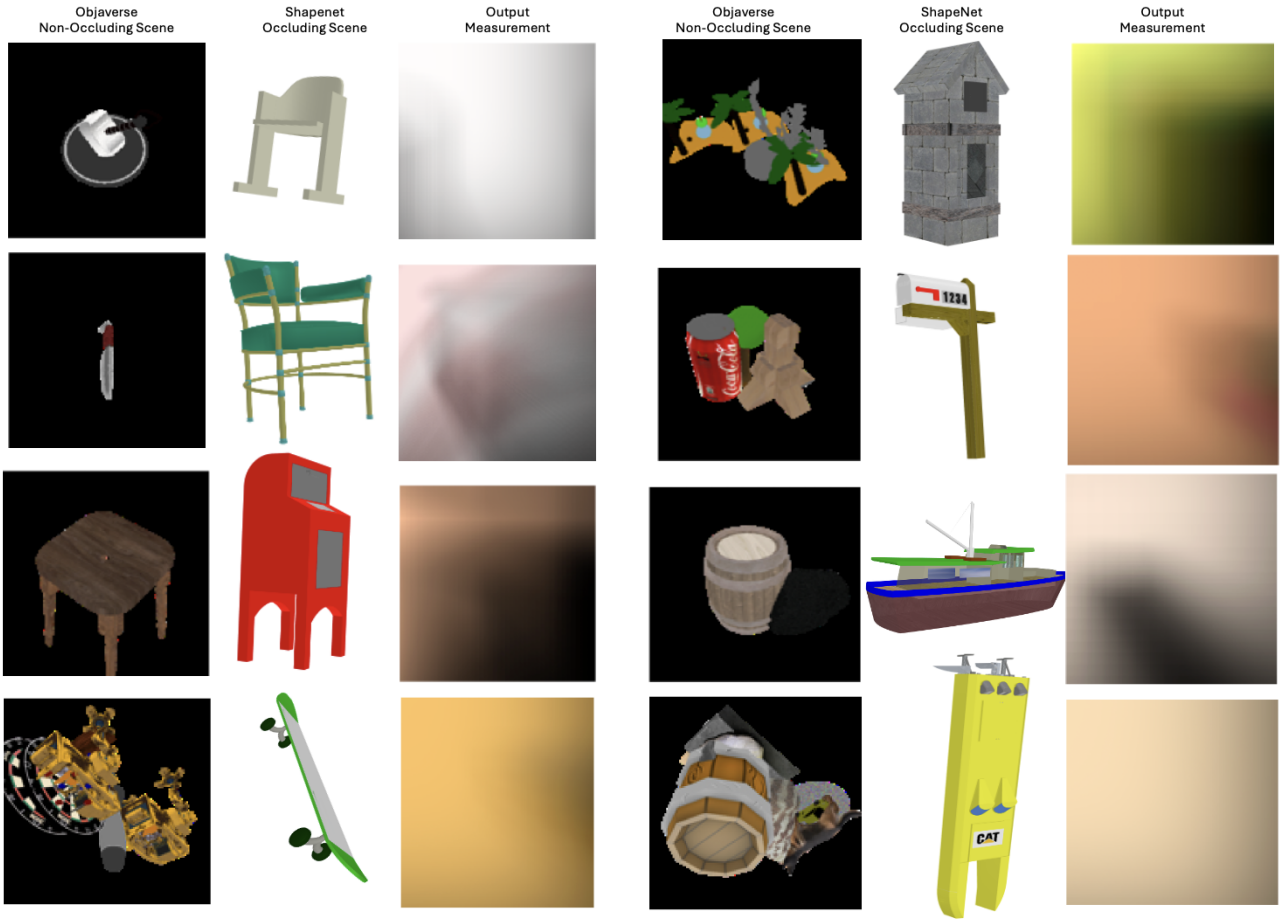

Figure 10. Examples of the Simulated Occluding and Non-Occluding Scene in the Training Dataset

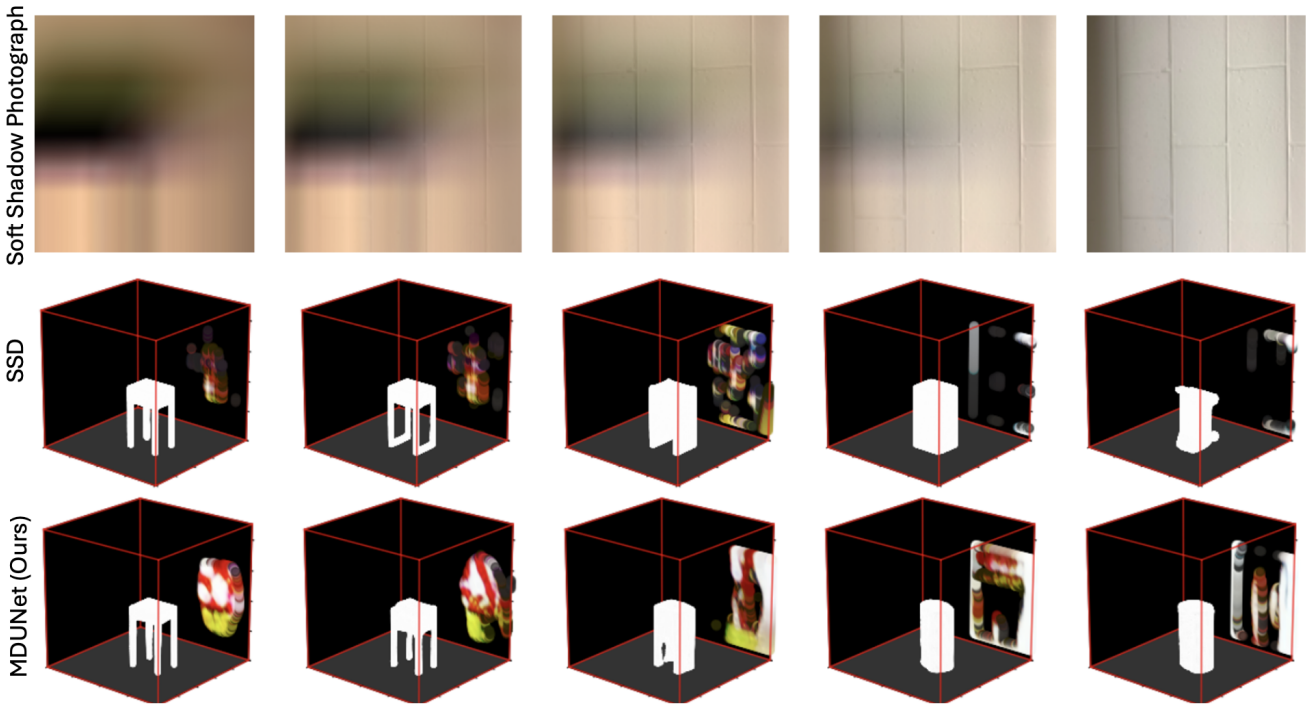

Figure 11. Impact of visible wall texture

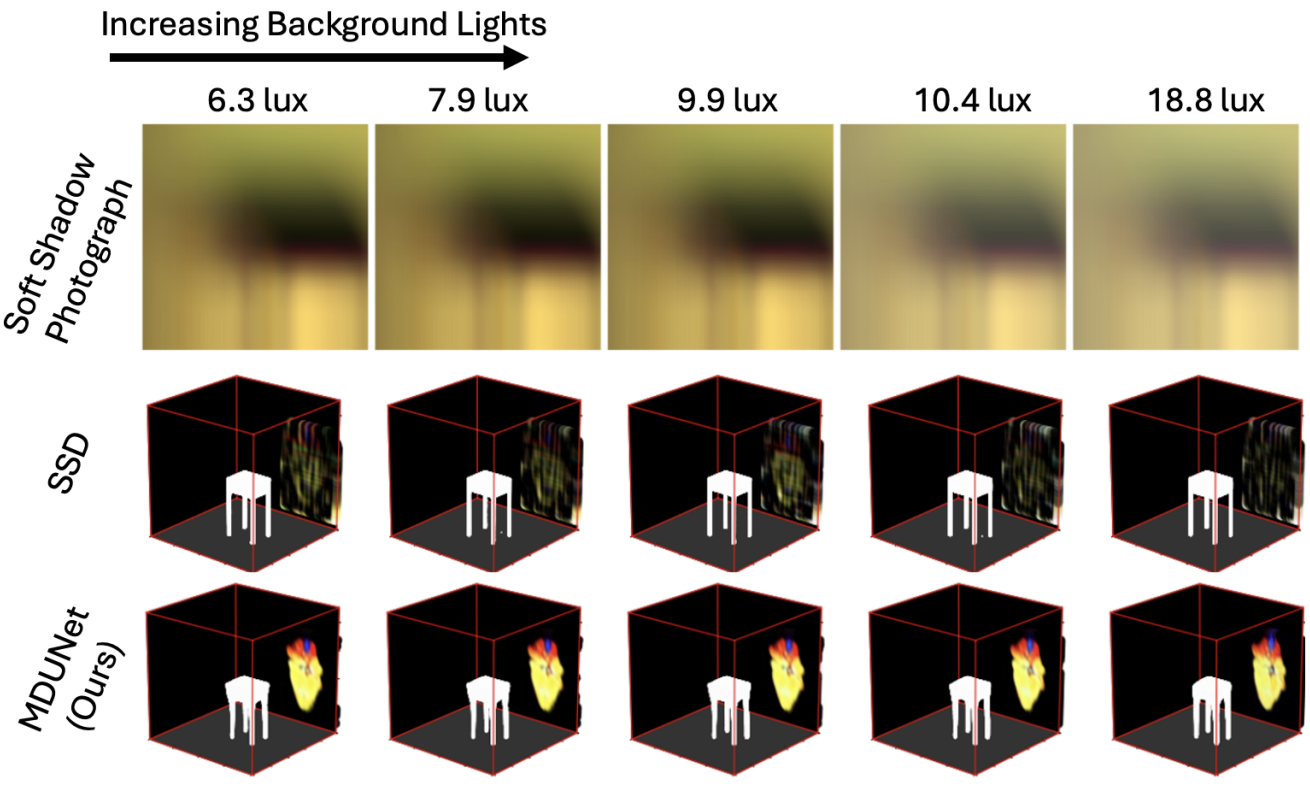

Figure 12. Impact of varying Background Lights