Neural Multi-View Self-Calibrated Photometric Stereo without Photometric Stereo Cues

Supplementary Material

This supplementary material provides implementation details of our method in Sec. 6, real-world data collection details in Sec. 7, scene normalization details in Sec. 8, and presents a visualization of the full BRDF map in Fig. 14.

6. Implementation Details

6.1. Network Architecture

As shown in Fig. 2, our forward rendering pipeline comprises three MLPs: the spatial MLP, the BRDF MLP, and the shadow MLP. We detail each MLP's architecture below and visualize them in Fig. 13.

Spatial MLP. The spatial MLP takes as input the hash-encoded point features concatenated with the point's coordinates and outputs a 64-dimensional vector. The first entry of the output vector represents the signed distance g, while the remaining entries serve as the BRDF latent vector $\mathbf{b}$. The spatial MLP consists of one hidden layer with 64 channels using the softplus activation. The output layer does not employ any activation function.

Regarding hash encoding, we set the base resolution to 32 and use 14 levels with 2 entries per level, yielding a 28-dimensional feature vector.

BRDF MLP. The BRDF MLP inputs the BRDF latent vector $\mathbf{b}(\mathbf{x})$ and the angularly encoded normal-view-light directions, producing a 3-dimensional output accounting for RGB channels. It comprises two hidden layers, each with 64 channels and ReLU activations. The output layer also employs a ReLU activation to ensure non-negative BRDF values.

Shadow MLP. The shadow MLP takes as input the surface point's latent vector $\mathbf{b}(\mathbf{x}')$, the viewing direction $\mathbf{v}$, and the volume-rendered shadow value s. We apply a third-degree spherical harmonics encoding to the viewing direction. The shadow MLP consists of two hidden layers with 64 channels and ReLU activations. The output layer uses a sigmoid activation to ensure the output shadow factors are within the range $[0, 1]$.

6.2. Loss Functions

As described in Sec. 3.3, the loss function comprises three terms: color loss, mask loss, and Eikonal loss. The color loss is defined in Eq. (16) of the main paper. Here, we provide details for the remaining two terms.

Mask loss. Given a batch of sampled pixels $\mathcal{P}$ from input images, we render the accumulated opacities $m(\mathbf{p})$ and compare them against the input binary mask values $\hat{m}(\mathbf{p})$ using the binary cross-entropy (BCE) loss:

$$\mathcal{L}_{\text{mask}} = \sum_{\mathbf{p}\in\mathcal{P}} \text{BCE}\left(m(\mathbf{p}), \hat{m}(\mathbf{p})\right). \tag{17}$$

Eikonal loss. Along the viewing rays cast from the camera centers, we sample a set of points $\mathcal{X}$ via ray marching. The Eikonal loss encourages the gradient norms of the SDF at all sample points to be 1:

$$\mathcal{L}_{\text{Eikonal}} = \sum_{\mathbf{x}\in\mathcal{X}} \left(\|\nabla g(\mathbf{x})\|_2 - 1\right)^2. \tag{18}$$

Since we only sample points near the surface, the Eikonal loss enforces a unit spatial gradient only in the vicinity of the zero level set of the SDF.

6.3. Optimization

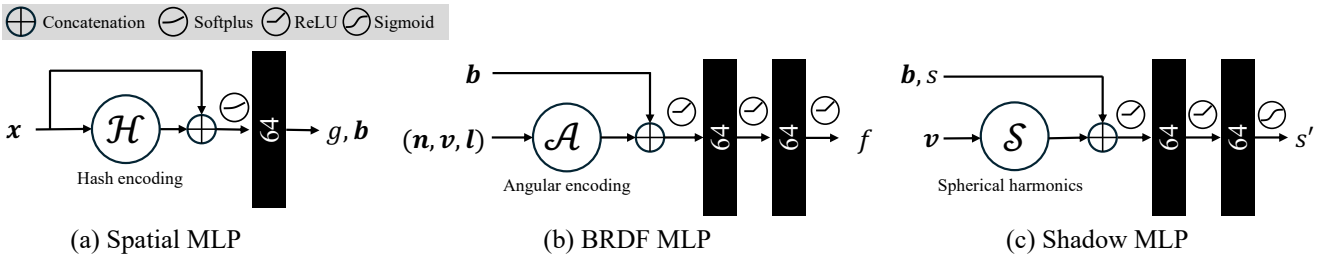
We implement our method using the PyTorch Lightning framework. Tiny-cuda-nn [36] and NerfAcc [28] are used to accelerate ray marching and volume rendering. Training takes about $20\,\text{min}$ per DiLiGenT-MV object and about $100\,\text{min}$ per self-collected object on an NVIDIA A100 GPU.

All terms in the loss function are weighted equally. We use the AdamW optimizer [33] with an initial learning rate of 1×10^{-2} for the parameters of the spatial MLP and BRDF MLP and 1×10^{-3} for the remaining parameters. All parameters are jointly trained for $20{,}000$ steps on DiLiGenT-MV [27] objects and $100{,}000$ steps on our self-collected data. In each step, we randomly sample $4{,}096$ rays for rendering.

Using NerfAcc [28], we update an occupancy grid every 16 steps during optimization to skip ray marching in empty regions (grids with occupancy values within a tiny threshold). For shadow ray marching, we set $t_{\text{near}} = 1 \times 10^{-2}$ and $t_{\text{far}} = 0.5$, and uniformly sample 64 points along the shadow ray segment.

6.4. Evaluation Metrics

L2 Chamfer Distance (CD). CD quantifies the similarity between two sets of points by calculating the average closest point distance from each point in one set to the other set.



(a) Spatial MLP (b) BRDF MLP (c) Shadow MLP

Figure 13. Our MLP architectures.

Given two point sets χ_1 and χ_2, CD is defined as

$$\begin{aligned} \text{CD} = & \frac{1}{|\chi_1|} \sum_{\mathbf{x}_1 \in \chi_1} \min_{\mathbf{x}_2 \in \chi_2} \|\mathbf{x}_1 - \mathbf{x}_2\|_2 \\ & + \frac{1}{|\chi_2|} \sum_{\mathbf{x}_2 \in \chi_2} \min_{\mathbf{x}_1 \in \chi_1} \|\mathbf{x}_2 - \mathbf{x}_1\|_2. \end{aligned} \tag{19}$$

Here, $\|\cdot\|_2$ denotes the Euclidean distance, and $|\cdot|$ represents the cardinality of the point set. In this work, we measure Chamfer Distance in millimeters (mm). A lower Chamfer Distance indicates a greater similarity between the two point sets.

Following SuperNormal [8], we compute the Chamfer Distance for points that are visible from the input views. To this end, we cast rays for pixels inside the foreground mask from all captured views and find their first intersection with the reconstructed or GT meshes.

Mean Angular Error (MAE). MAE measures the average angular difference between corresponding unit vectors. Given two sets of unit vectors $\{\mathbf{n}_i\}$ and $\{\hat{\mathbf{n}}_i\}$, the mean angular error is defined as

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} \arccos\left(\mathbf{n}_i^\top \hat{\mathbf{n}}_i\right). \tag{20}$$

In this work, MAE is measured in degrees. A lower MAE indicates a smaller average angular discrepancy between the vectors, signifying higher accuracy in normal or light direction estimation.

Since normal maps are rendered in multiple views, we apply Eq. (20) to normal vectors collected from all rendered views within the foreground masks. Because rotation transformations preserve angles, computing MAE using either world-space or camera-space normal maps yields the same value.

Peak Signal-to-Noise Ratio (PSNR). PSNR measures the similarity between the rendered images and the captured images. It is defined based on the mean squared error (MSE) as:

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right). \tag{21}$$

Here, MAX is the maximum possible pixel value of the image (1 in our case), and MSE is the mean squared error between the rendered and captured images. Higher PSNR values indicate better image quality.

Scale-invariant Mean Squared Error (SI-MSE). As the light intensities can only be estimated up to a scale, we follow previous work [26] to use the scale-invariant relative error defined as:

$$\text{SI-MSE} = \frac{1}{M} \sum_{j=1}^{M} \frac{|s_e \hat{e}_j - e_j|}{e_j}, \tag{22}$$

where $\hat{e}_j$ and e_j denote the estimated and ground truth light intensities of the j-th light, respectively. The scale factor s_e is computed by solving the least squares problem:

$$s_e = \arg\min_{s_e} \sum_{j=1}^{M} \left(s_e \hat{e}_j - e_j\right)^2. \tag{23}$$

A lower scale-invariant relative error indicates a more accurate estimation of the light intensities.

7. Real-World Data Collection

7.1. Capture Setup Details

As shown in Fig. 10, we prepare six strobe lights that emit point flashlights and a Sony ILCE-7RM5 camera equipped with a telephoto zoom lens. Both the camera and the strobes are mounted on tripods to ensure that each light remains fixed relative to the camera. Since our method does not require a light to be static to other lights, preparing M light sources is equivalent to repositioning a single source $M-1$ times during capture.

Following the setup of DiLiGenT-MV [27], we place the stobes approximately $1\,\text{m}$ away from centimeter-scale objects, so that the point flashlight can be safely approximated

as directional. To ensure sufficient image coverage, we adjust the focal length between $70\,\mathrm{mm}$ and $100\,\mathrm{mm}$ depending on the object size. The aperture is set between F/11 and F/22 to achieve a large depth of field, ensuring the entire object remains in focus. The ISO is set between 100 and 200 to minimize sensor noise.

Capture is conducted indoors. While a darkroom is ideal for OLAT acquisition, it is not a strict requirement as long as the flash emits a burst of light that is momentarily much stronger than the ambient illumination. Under such lighting conditions, a short camera exposure (e.g., 1/200 sec) effectively reduces the influence of ambient light to a negligible level.

Figure 15 visualizes the distribution of camera viewpoints. A motorized turntable rotates the object in 15° increments, yielding 24 uniformly distributed multi-view OLAT images per strobe light. Each time we switch to a different light, we adjust the camera height and elevation angles to introduce more viewpoint variations. In total, we capture 144 OLAT images per object (6L24V) at a resolution of 9504×6336. Under our setting, the real-world resolution of each pixel is approximately $0.04\,\mathrm{mm}$.

7.2. Image Preprocessing

The DSLR camera produces a pair of raw and JPEG images per exposure. The raw images are unprocessed and directly fed into our algorithm. For camera calibration, we use the JPEG images and employ RealityCapture [1] due to its efficiency and robustness. This yields a single 3×4 world-to-image projection matrix per view, encoding both the world-to-camera transformation and the perspective camera intrinsics.

Multi-view foreground masks are generated using SAM2 [42], which supports efficient and automated cross-view mask propagation. With a few mouse clicks on a single image, SAM2 [42] segments the corresponding mask and propagates it to other views in seconds. To facilitate more reliable foreground segmentation, we apply gamma correction to the JPEG images to enhance contrast between the foreground object and the background.

Finally, given the camera parameters and multi-view foreground masks, we perform scene normalization such that the target object is enclosed within a unit sphere. This process is detailed in the next section.

8. Scene Normalization

Scene normalization applies a global scaling and translation to world coordinates such that the target object is bounded within a unit sphere [51]. This normalization facilitates the training of the neural SDF and is conducted before reconstruction. For clarity, **we refer to the normalized world coordinates as the object coordinates**, where the unit sphere is centered at the origin. In the following, Sec. 8.1 describes the coordinate system transformations involved in our rendering pipeline, and Sec. 8.2 provides a robust scene normalization method based on camera parameters and multi-view foreground masks.

8.1. Coordinate System Transformations

We define the object coordinates to differ from the world coordinates by an isotropic scale and translation without rotational motion. Specifically, for a point $\mathbf{x}^{\mathrm{O}} \in \mathbb{R}^3$ in object coordinates, its world coordinates is given by $\mathbf{x}^{\mathrm{W}} = s\mathbf{x}^{\mathrm{O}} + \mathbf{d}$, where $s \in \mathbb{R}_+$ is the object-to-world scale and $\mathbf{d} \in \mathbb{R}^3$ is the object-to-world translation[3]. In homogenous coordinates, the object-to-world transformation $\mathbf{T}_{\mathrm{O2W}}$ is

$$\tilde{\mathbf{x}}^{\mathrm{W}} = \underbrace{\begin{bmatrix} s\mathbf{I} & \mathbf{d} \\ \mathbf{0}^\top & 1 \end{bmatrix}}_{\mathbf{T}_{\mathrm{O2W}}} \tilde{\mathbf{x}}^{\mathrm{O}}. \tag{24}$$

With the object space, three transformations are involved to project an object-space point onto the i-th image plane: Object space $\rightarrow$ world space $\rightarrow$ i-th camera space $\rightarrow$ i-th image plane, which can be formally described as

$$z_i \tilde{\mathbf{u}}_i = \mathbf{K}_i \mathbf{T}_{\mathrm{W2C}_i} \mathbf{T}_{\mathrm{O2W}} \tilde{\mathbf{x}}^{\mathrm{O}} \tag{25}$$

$$= \mathbf{K}_i \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \end{bmatrix} \begin{bmatrix} s\mathbf{I} & \mathbf{d} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tilde{\mathbf{x}}^{\mathrm{O}}. \tag{26}$$

Here, $\tilde{\mathbf{u}}_i$ is the homogeneous coordinates of the pixel coordinates, z_i is the depth of the pixel, $\mathbf{K}_i$ is the 3×3 intrinsic matrix of the i-th camera, and $\mathbf{R}_i \in SO(3)$ and $\mathbf{t}_i \in \mathbb{R}^3$ represent the world-to-camera rotation and translation, respectively.

Camera position in object coordinates. In volume rendering, camera position and viewing directions are required to determine the ray from which we sample points. In object coordinates, the camera position $\mathbf{c}_i^{\mathrm{O}}$ can be obtained from its world coordinates $\mathbf{c}_i^{\mathrm{W}}$ as

$$\mathbf{c}_i^{\mathrm{O}} = (\mathbf{c}_i^{\mathrm{W}} - \mathbf{d})/s \quad \text{with} \quad \mathbf{c}_i^{\mathrm{W}} = -\mathbf{R}_i^\top \mathbf{t}_i. \tag{27}$$

Equivalently, $\mathbf{c}_i^{\mathrm{O}}$ can be obtained as the null vector of object-to-camera transformation since it transforms the camera position to the origin in camera coordinates:

$$\mathbf{T}_{\mathrm{W2C}_i} \mathbf{T}_{\mathrm{O2W}} \tilde{\mathbf{c}}_i^{\mathrm{O}} = \mathbf{0}. \tag{28}$$

Ray directions in object coordinates. The scale and translation do not affect directional vectors. Therefore, the viewing directions in object coordinates stay the same as in world coordinates:

$$\mathbf{v}^{\mathrm{O}} = \mathbf{v}^{\mathrm{W}} = \mathbf{R}_i^\top \mathbf{K}_i^{-1} \tilde{\mathbf{u}}. \tag{29}$$

The same applies to normal and light directions.

[3]The symbol s denotes the object-to-world scale in this section only, whereas it is referred to the shadow factor in the main paper.

Mesh vertices in world coordinates. Since the neural SDF is trained in object coordinates, extracting its zero-level set by marching cubes yields a mesh with vertices in object coordinates. Converting the mesh to world coordinates is realized by

$$\mathbf{p}^{\mathrm{W}} = s\mathbf{p}^{\mathrm{O}} + \mathbf{d}, \tag{30}$$

where $\mathbf{p}^{\mathrm{O}}$ indicates the object coordinates of a mesh vertex, and Eq. (30) is applied to all vertices.

8.2. O2W Scale and Translation Estimation

Estimating the object-to-world scale and translation before reconstruction requires prior knowledge about the scene. For object-centric reconstruction, the object space origin can be defined as the point closest to the camera principal axes of all views [52]. However, this may not position the object in the center of the unit sphere well if the principal axes of the surrounding cameras do not point to the target object. For large-scale scene reconstruction, sparse 3D points reconstructed by structure-from-motion are used [29], and manual effort is required to segment the scene of interest.

In this work, we use the known camera parameters and foreground masks, which are readily available as input to our method, to normalize the scene. Given that the object is assumed to lie within a unit sphere in object space, the projection of this unit sphere onto each image should ideally encompass the foreground pixels across all views. Based on this constraint, we use a two-step process to estimate the object-to-world translation and scale.

O2W translation estimation. We expect the object to be centered at the origin of the object space. To this end, we estimate the O2W translation vector $\mathbf{d}$ such that the object space origin lies as close as possible to the foreground center-of-mass rays across all views. Let $\mathbf{o}_i + t\mathbf{v}_i$ be the ray in world coordinates passing through the center of mass of the foreground region in the i-th view, where $\mathbf{o}_i$ is the camera center and $\mathbf{v}_i$ is the corresponding viewing direction. In world coordinates, the origin of the object space is located at $\mathbf{d}$. The squared distance from $\mathbf{d}$ to the ray is given by

$$d^2(\mathbf{d}, \mathbf{o}_i + t\mathbf{v}_i) = \mathbf{d}^\top \mathbf{V}_i \mathbf{d} - 2\mathbf{o}_i^\top \mathbf{V}_i \mathbf{d} + \mathbf{o}_i^\top \mathbf{V}_i \mathbf{o}_i, \tag{31}$$

where $\mathbf{V}_i = \mathbf{I} - \mathbf{v}_i \mathbf{v}_i^\top$. The O2W translation is then obtained by minimizing the sum of squared distances across all views:

$$\mathbf{d}^* = \underset{\mathbf{d}}{\operatorname{argmin}} \sum_i d^2(\mathbf{d}, \mathbf{o}_i + t\mathbf{v}_i). \tag{32}$$

The objective Eq. (32) is quadratic and convex, and thus attains a global optimum at the critical point where the gradient takes 0. Setting the gradient of Eq. (32) to zero yields the normal equation:

$$\mathbf{V}^\top \mathbf{V} \mathbf{d} = \mathbf{V}^\top \mathbf{b} \tag{33}$$

$$\text{with} \quad \mathbf{V} = \sum_i \mathbf{V}_i, \quad \mathbf{b} = \sum_i \mathbf{V}_i \mathbf{o}_i. \tag{34}$$

We solve Eq. (34) using a least-squares solver.

O2W scale estimation. Once the O2W translation $\mathbf{d}^*$ is determined, we then estimate the O2W scale s such that the projected unit sphere fully encloses the foreground regions in all views. This is enforced by ensuring that the sum of the projected sphere areas of all views is greater than a factor k times the sum of all foreground areas:

$$s^* = \underset{s}{\operatorname{argmin}} \left(\sum_i A_i(s) \right) \geq k \sum_i \hat{A}_i. \tag{35}$$

$\hat{A}_i$ is the foreground area, and $A_i(s)$ is the projected sphere area in the i-th view. To analytically compute $A_i(s)$, we make two approximations: (1) The projection of a 3D sphere onto the image plane is approximately circular, and (2) the radius r_i of this projected circle can be approximated using the principle of similar triangles:

$$\frac{r_i}{s} \approx \frac{f_i}{z_i}, \tag{36}$$

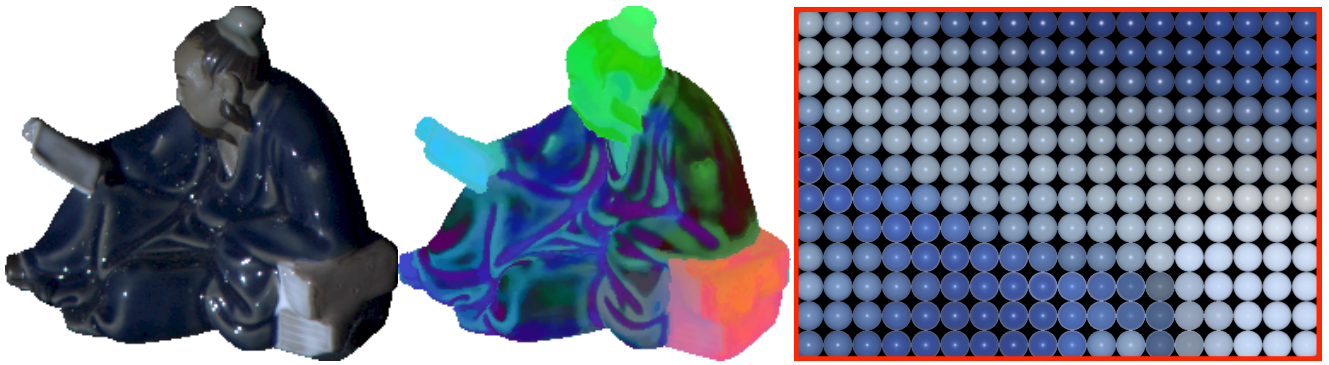
where f_i is the focal length of i-th viewpoint, z_i is the depth of object-space origin in i-th camera coordinates, *i.e.*, $z_i = (\mathbf{R}_i \mathbf{d}^* + \mathbf{t}_i)_z$. With the approximations, the projected area of the unit sphere in the i-th view is given by

$$A_i(s) = \pi r_i^2 \approx \frac{\pi s^2 f_i^2}{z_i^2}. \tag{37}$$

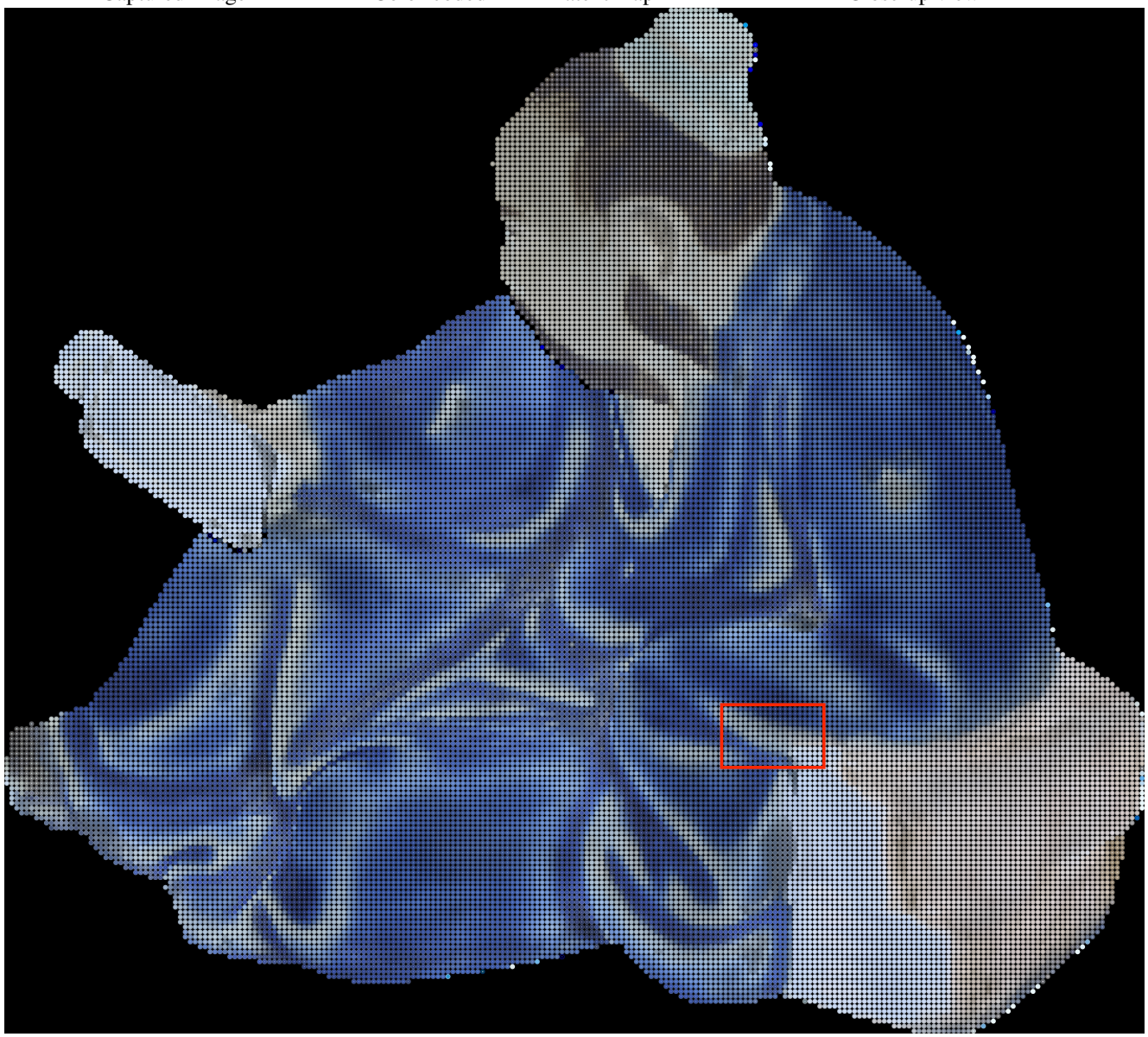
Substituting Eq. (37) into Eq. (35) yields a closed-form solution for the O2W scale:

$$s = \sqrt{\frac{k \sum_i \hat{A}_i}{\pi \sum_i \left(\frac{f_i^2}{z_i^2} \right)}}. \tag{38}$$

Empirically, we find that a scaling factor of $k = 5$ works well for all DiLiGenT-MV [27] scenes and our captured objects, as shown in Fig. 16.



Captured image | Color-coded BRDF latent map | Close-up view



Full BRDF map

Figure 14. **BRDF map.** For each pixel in the input image, we retrieve its corresponding surface point from the spatial MLP and render its BRDF on a sphere under a colocated camera and light.

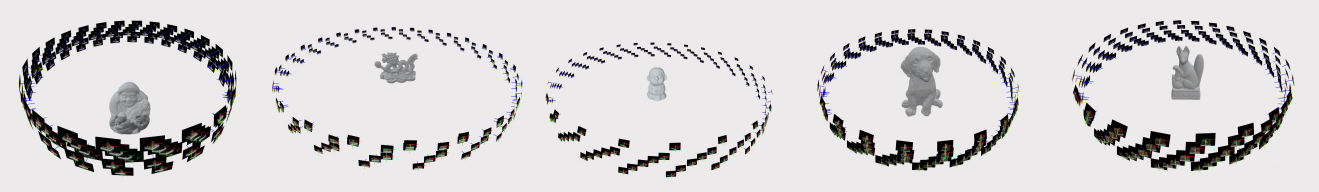


Figure 15. Visualization of camera positions with respect to the object. Our setup does not contain aligned viewpoints across lighting, and images on the same elevation are captured under the same light source. Meshes are produced by RealityCapture [1] and enlarged for improved visibility.

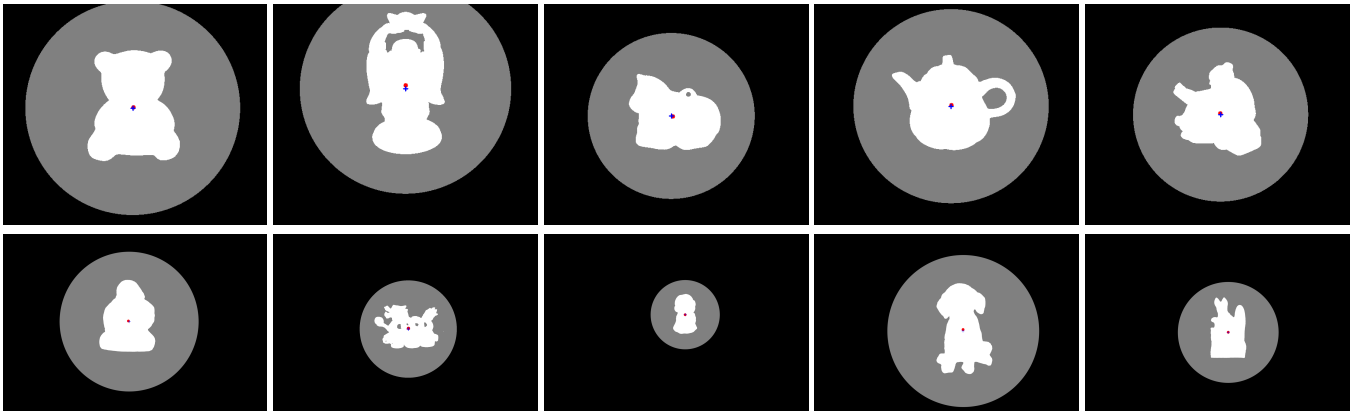


Figure 16. **Visualization of scene normalization results.** (Top row) DiLiGenT-MV [27] scenes. (Bottom row) Our self-captured scenes. White pixels indicate foreground regions, gray pixels denote the projected unit sphere, red circles mark the center of mass of the foreground regions, and blue crosses indicate the centers of the projected unit sphere.