Supplementary: Complex-Valued Holographic Radiance Fields

YICHENG ZHAN, University College London, United Kingdom
DONG-HA SHIN, Pohang University of Science and Technology (POSTECH), South Korea
SEUNG-HWAN BAEK, Pohang University of Science and Technology (POSTECH), South Korea
KAAN AKŞIT, University College London, United Kingdom

ACM Reference Format:
Yicheng Zhan, Dong-Ha Shin, Seung-Hwan Baek, and Kaan Akşit. 2024. Supplementary: Complex-Valued Holographic Radiance Fields. 1, 1 (March 2024), 19 pages. https://doi.org/XXXXXXX.XXXXXXX

A Hardware Image

Fig. 1 shows the photograph of the holographic display prototype we used in this paper. The optical path of our display prototype begins with a laser light source (LASOS MCS4), which integrates three individual laser lines. The emitted light from a single-mode fibre is collimated using a Thorlabs LA1708-A plano-convex lens with a 200 mm focal length. This linearly polarized, collimated beam is then directed by a beamsplitter (Thorlabs BP245B1) toward our phase-only Spatial Light Modulator (SLM), the Jasper JD7714 (2400×4094, 3.74 μm). The modulated beam subsequently passes through a lens system comprising Thorlabs LA1908-A and LB1056-A, with focal lengths of 500 mm and 250 mm, respectively. Following this, a pinhole aperture (Thorlabs SM1D12) is positioned at the focal plane of the lenses. Finally, we capture the holographic reconstructions using a lensless image sensor (Point Grey GS3-U3-23S6M-C USB 3.0), which is mounted on an X-stage (Thorlabs PT1/M) with a travel range of 0 to 25 mm and a positioning precision of 0.01 mm.

B Wave Propagation

In this section, we provide additional insights into the implementation of our proposed complex-valued holographic radiance field.

B.1 Wave Propagation From a Single 3D Gaussian Primitive

Each Gaussian $\mathcal{G}_n$ can be considered as a small volume in Ω that emits a coherent wavefield. Intuitively, we may approximate the total field from $\mathcal{G}_n$ at hologram pixel P_i's coordinate (x, y) by an integral of infinitesimal point contributions at the surface of the Gaussian. **To simplify the problem, if we only consider $\mathcal{G}_n$'s center at $\mathbf{x}_n = (X_n, Y_n, Z_n)$**, the distance D_n from the $\mathcal{G}_n$ to the hologram pixel at (x, y) is

$$D_n = \sqrt{(x - X_n)^2 + (y - Y_n)^2 + Z_n^2} \tag{1}$$

For a point light source emitting spherical waves, the phase at the hologram pixel is

$$\boldsymbol{\varphi}_n^{\text{prop}} = \frac{2\pi}{\lambda} D_n \mod 2\pi \tag{2}$$

Under paraxial approximation, phase can be described using a quadratic function

$$\boldsymbol{\varphi}_n^{\text{approx}} = \frac{2\pi}{\lambda}\left(Z_n + \frac{(x - X_n)^2 + (y - Y_n)^2}{2Z_n}\right) \mod 2\pi \tag{3}$$

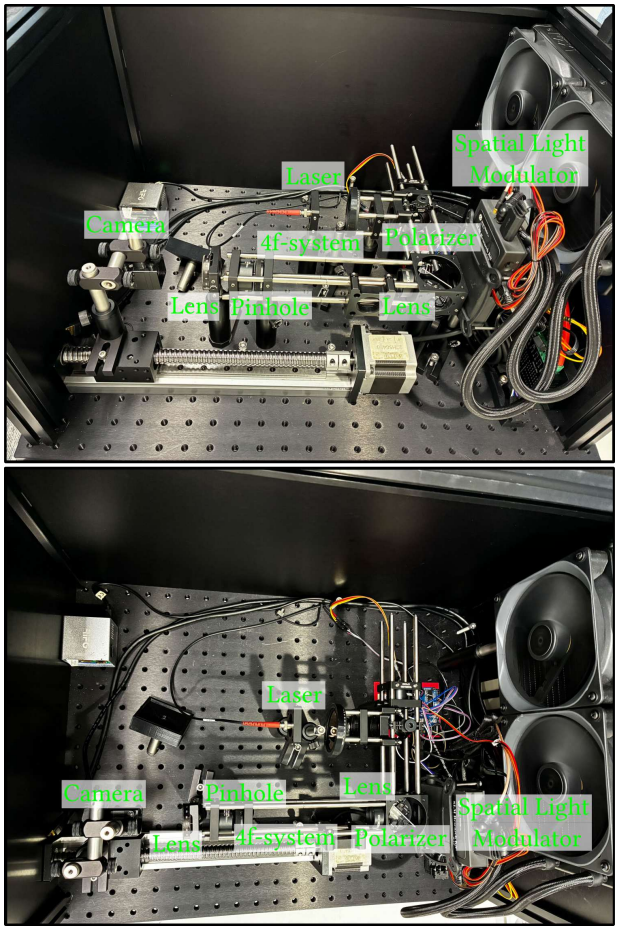

Fig. 1. The photograph of the holographic display prototype (Jasper JD7714) used in evaluating holograms generated by our model.

Authors' addresses: Yicheng Zhan, UCABY83@ucl.ac.uk, University College London, Gower Street, London, United Kingdom, WC1E 6BT; Dong-Ha Shin, 0218sdh@gmail.com, Pohang University of Science and Technology (POSTECH), 77 Cheongam-ro, Nam-gu, Pohang-si, South Korea, 37673; Seung-Hwan Baek, shwbaek@postech.ac.kr, Pohang University of Science and Technology (POSTECH), 77 Cheongam-ro, Nam-gu, Pohang-si, South Korea, 37673; Kaan Akşit, University College London, Gower Street, London, United Kingdom, k.aksit@ucl.ac.uk.

XXXX-XXXX/2024/3-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

The resulting complex field from Gaussian $\mathcal{G}_n$ at the hologram pixel is

$$U_n = A_n \exp(j\boldsymbol{\varphi}_n^{\text{prop}}), \quad A_n = \mathbf{c}_n\, \boldsymbol{\alpha}_n \tag{4}$$

where A_n includes the Gaussian's $\boldsymbol{\alpha}_n$ weighted amplitude, and $\mathbf{c}_n$ represents the inherent wave amplitude of the Gaussian. The full complex field integration of $\mathcal{G}_n$ can be derived by expanding this point response to the Gaussian's spatial extent $\mathbf{S}_n$ and rotation $\mathbf{R}_n$ by densely discretizing the surface of the Gaussian. This yields a volume of wavefront footprint recorded at the hologram plane P. Without considering the occlusion, the total field at the hologram pixel can be defined as:

$$U^{\text{total}} = \sum_{n=1}^{N} U_n \tag{5}$$

One may average over the sub-samples to compute the field at hologram pixel P_i, or keeping them explicit for higher accuracy modeling.

B.2 Occlusion Aware Field Recording.

In a pure intensity world, to account for fore-ground occlusion, the frontmost Gaussian along the line of sight tends to occlude anything behind it, and people apply alpha-blending to represent transparent pixels. For simulating physical wave propagation in a hologram, occlusion is subtler—each "surface" can block or partially block wavefronts behind it. One approach to accurately model occlusion is to reconstruct a triangle surface mesh by connecting the neighboring points of Gaussians and performing ray casting from each point to the hologram plane P. The wavefronts carried by rays that hit the surface mesh are excluded from the hologram calculation. A binary visibility mask can be generated from this process to mimic foreground-occlusion at the hologram plane. However, both point-based wave simulation and ray-triangle detection of the entire scene are computationally expensive, making them a less favorable choice in real-time inference. As illustrated in Fig. 2, in our implementation, we use alpha-blending with transmittance to efficiently approximate occlusion effects, each 3D Gaussian $\mathcal{G}_n$ can be considered as a small volume in Ω that emits a coherent wavefield, rather than the more computationally expensive ray-casting approach described above.

C Training Camera Viewpoint Sampling

During training, camera viewpoints are sampled differently depending on the dataset characteristics to ensure the learned scene representation supports consistent rendering across viewpoints within each dataset's coverage.

For the NeRF Synthetic dataset, we randomly sample camera poses facing towards the object within the Blender coordinate system, maintaining the same radial distance distribution as the original dataset while randomizing the azimuth and elevation angles. This generates a continuous distribution of training viewpoints around the object.

For the LLFF dataset, we directly use the existing camera viewpoints provided in the dataset, as the captured views already provide sufficient coverage of the indoor scenes with manageable motion parallax.

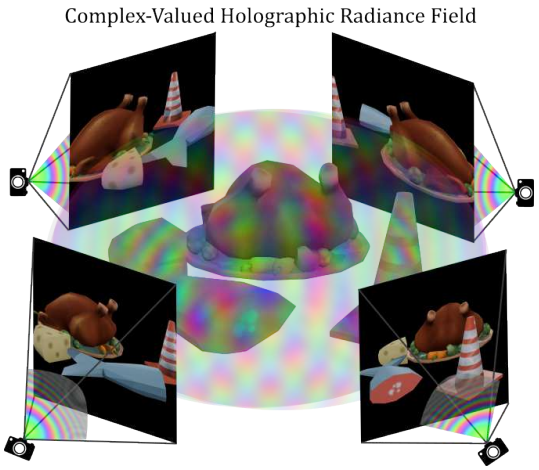

Fig. 2. Our method assumes the radiance field is a wave emitter, instead of a ray tracer.

For the Mip-NeRF 360 dataset, we employ a two-stage approach due to complex field discontinuities challenges arising from extreme motion parallax in the original wide-baseline captures. First, we train an intensity-based 3DGS scene using the provided dataset images. We uniformly sample 480 camera views covering a full 360° rotation, corresponding to 0.75° angular spacing for training, and evaluate novel-view rendering on the intermediate unseen viewpoints between sampled angles. We then use this trained 3DGS scene as a reference to randomly sample camera viewpoints facing towards the scene center, similar to the NeRF Synthetic sampling strategy. As a way round method, this approach generates training views with reduced inter-view motion parallax in the near scene compared to the original Mip-NeRF 360 captures, mitigating the severe complex field discontinuities that would otherwise cause substantial image quality degradation, but it does not help with the far field.

D Differentiable Complex-Valued Gaussian Rendering

In this section, we provide the pseudo code and a detailed exposition of both the forward computation and the backward gradient derivation for our Differentiable Complex-Valued Rasterizer.

D.1 Notation

- $\mathbf{x}_n \in \mathbb{R}^3$ - 3D Gaussian center positions in world space
- $\mathbf{x}_n^{\text{proj}} \in \mathbb{R}^2$ - Projected 2D Gaussian centers in screen space
- μ_x, μ_y - Components of the 2D mean position
- $\Sigma \in \mathbb{R}^{3\times3}$ - 3D covariance matrix in world space
- $\Sigma' \in \mathbb{R}^{2\times2}$ - 2D covariance matrix in screen space
- Σ'^{-1}_{ij} - Elements of the inverse 2D covariance matrix (where $i, j \in \{0, 1\}$)
- $d_x = x - \mu_x, d_y = y - \mu_y$ - Distance from pixel to Gaussian center
- $\boldsymbol{\rho}_{n,l}$ - Plane assignment probability of Gaussian n for plane l (controls contribution to each rendering plane)
- $\boldsymbol{\varphi}_n$ - Phase value for Gaussian n (controls complex field phase)

- $\boldsymbol{\alpha}_n$ - Opacity value for Gaussian n
- $\boldsymbol{\alpha}_n^{\text{base}}$ - Base opacity value before applying the Gaussian function
- T_n - Accumulated transmittance at Gaussian n (product of previous transparency values)
- z_{norm} - Normalized depth value between 0 and 1

D.2 Optimization and Densification Algorithm

Our optimization and densification algorithms extend the standard 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] approach to handle complex-valued parameters and multi-plane assignments. The key modifications include phase parameter optimization, plane probability learning with Straight-Through Estimator (STE), and complex-valued loss computation as summarized in Algorithm 1.

Algorithm 1 Optimization and Densification for Complex-Valued 3DGS

Require: w, h: width and height of the training images
Require: L: number of depth planes for multi-plane rendering
$M \leftarrow$ SfM Points ▷ Positions
$\mathbf{S}, \mathbf{C}, \boldsymbol{\alpha} \leftarrow$ InitAttributes() ▷ Scales, Amplitudes, Opacities
$\boldsymbol{\varphi} \leftarrow$ InitPhase() ▷ Phase Parameters
$\boldsymbol{\rho}' \leftarrow$ InitPlaneProbs(L) ▷ Plane Assignment Logits
$i \leftarrow 0$ ▷ Iteration Count
while not converged **do**
  $V, I \leftarrow$ SampleTrainingView() ▷ Camera V and Image
  $\boldsymbol{\rho} \leftarrow \text{STE}(\boldsymbol{\rho}')$ ▷ Hard Assignment via STE
  $U_{\text{complex}} \leftarrow \text{ComplexRasterize}(M, \mathbf{S}, \mathbf{C}, \boldsymbol{\alpha}, \boldsymbol{\varphi}, \boldsymbol{\rho}, V)$ ▷ Alg. 2
  $P \leftarrow \text{PropagateToHologram}(U_{\text{complex}})$ ▷ Multi-plane ASM
  $I_{\text{recon}} \leftarrow \text{BackPropagate}(P)$ ▷ Reconstruct Images
  $\mathcal{L} \leftarrow \text{ComplexLoss}(I, I_{\text{recon}})$ ▷ $\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{SSIM}}$
  $M, \mathbf{S}, \mathbf{C}, \boldsymbol{\alpha}, \boldsymbol{\varphi}, \boldsymbol{\rho}' \leftarrow \text{Adan}(\nabla \mathcal{L})$ ▷ Backprop & Step
  if IsRecentIteration() **then**
    for all Gaussians $(\mu, \Sigma, c, \alpha, \varphi, \rho')$ **in** $(M, \mathbf{S}, \mathbf{C}, \boldsymbol{\alpha}, \boldsymbol{\varphi}, \boldsymbol{\rho}')$ **do**
      if $\alpha < \epsilon$ **or** IsTooLarge(μ, Σ, c, α) **then** ▷ Pruning
        RemoveGaussian()
      end if
      if $\nabla_\mu \mathcal{L} > \tau_p$ **then** ▷ Densification
        if $\|\mathbf{S}\| > \tau_s$ **then** ▷ Over-reconstruction
          SplitGaussian($\mu, \Sigma, c, \alpha, \varphi, \rho'$)
        else ▷ Under-reconstruction
          CloneGaussian($\mu, \Sigma, c, \alpha, \varphi, \rho'$)
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while

The key extensions to the standard algorithm include: (1) Phase parameter $\boldsymbol{\varphi}$ initialization with random values in $[0, 2\pi)$; (2) Plane assignment probabilities $\boldsymbol{\rho}'$ initialized uniformly and converted to hard assignments via STE during forward pass; (3) Complex-valued rasterization producing multi-plane complex fields; (4) Hologram generation through multi-plane Angular Spectrum Method (ASM) propagation; (5) Phase-aware densification where both split and clone operations preserve phase relationships by adding small random perturbations to maintain coherent interference patterns.

D.3 Details of the Rasterizer

Our tile-based rasterizer extends the standard 3DGS approach to handle complex-valued rendering across multiple depth planes. The algorithm processes Gaussians in back-to-front order within each tile to ensure correct alpha blending for wave-optical effects, as detailed in Algorithm 2.

Algorithm 2 Complex-Valued Tile-Based Rasterization

Require: w, h: width and height of the image to rasterize
Require: $M, \mathbf{S}$: Gaussian means and covariances in world space
Require: $\mathbf{C}, \boldsymbol{\alpha}, \boldsymbol{\varphi}, \boldsymbol{\rho}$: Amplitudes, opacities, phases, and plane assignments
Require: V: view configuration of current camera
Require: L: number of depth planes
function ComplexRasterize($w, h, M, \mathbf{S}, \mathbf{C}, \boldsymbol{\alpha}, \boldsymbol{\varphi}, \boldsymbol{\rho}, V$)
  CullGaussians(M, V) ▷ Frustum Culling
  $M', \mathbf{S}' \leftarrow$ ScreenspaceGaussians($M, \mathbf{S}, V$) ▷ Transform
  $T \leftarrow$ CreateTiles(w, h) ▷ 16×16 Tiles
  $\mathcal{I}, \mathcal{K} \leftarrow$ DuplicateWithDepthKeys(M', T) ▷ Back-to-Front Keys
  $\mathcal{K}_s, \mathcal{I}_s \leftarrow$ SortByKeys($\mathcal{I}, \mathcal{K}$) ▷ Depth Sort
  $\mathcal{R} \leftarrow$ IdentifyTileRanges($T, \mathcal{K}_s$) ▷ Tile Ranges
  $U_{\text{real}}, U_{\text{imag}} \leftarrow$ InitComplexCanvas(L, w, h) ▷ Multi-plane Output
  for all Tiles t **in** T **do**
    for all Planes l **in** L **do**
      for all Pixels pix **in** t **do**
        $T_{\text{acc}} \leftarrow 1.0$ ▷ Transmittance
        $\text{real}_{\text{acc}}, \text{imag}_{\text{acc}} \leftarrow 0.0, 0.0$ ▷ Complex Accumulation
        range $\leftarrow$ GetTileRange($\mathcal{R}, t$)
        for g **in** range **do** ▷ Back-to-Front Order
          $w_{\text{plane}} \leftarrow \boldsymbol{\rho}_{g,l}$ ▷ Plane Assignment
          if $w_{\text{plane}} < \epsilon_{\text{plane}}$ **then**
            continue
          end if
          $\alpha_{\text{eff}} \leftarrow \boldsymbol{\alpha}_g \cdot \exp(-\text{power}_{g,\text{pix}}) \cdot w_{\text{plane}}$ ▷ Effective Alpha
          $\text{scale} \leftarrow \mathbf{C}_g \cdot \alpha_{\text{eff}} \cdot T_{\text{acc}}$ ▷ Complex Scale
          $\text{real}_{\text{acc}} \leftarrow \text{real}_{\text{acc}} + \text{scale} \cdot \cos(\boldsymbol{\varphi}_g)$ ▷ Real Component
          $\text{imag}_{\text{acc}} \leftarrow \text{imag}_{\text{acc}} + \text{scale} \cdot \sin(\boldsymbol{\varphi}_g)$ ▷ Imaginary Component
          $T_{\text{acc}} \leftarrow T_{\text{acc}} \cdot (1 - \alpha_{\text{eff}})$ ▷ Update Transmittance
          if $T_{\text{acc}} < \epsilon_T$ **then**
            break ▷ Early Termination
          end if
        end for
        $U_{\text{real}}[l, \text{pix}] \leftarrow \text{real}_{\text{acc}}$
        $U_{\text{imag}}[l, \text{pix}] \leftarrow \text{imag}_{\text{acc}}$
      end for
    end for
  end for
  return $U_{\text{real}} + j \cdot U_{\text{imag}}$
end function

The rasterizer incorporates several key innovations: (1) Back-to-front depth sorting using inverted depth keys to ensure proper alpha blending for wave propagation; (2) Multi-plane rendering where each Gaussian contributes to planes based on learned assignment probabilities $\boldsymbol{\rho}_{g,l}$; (3) Complex field accumulation using amplitude-weighted trigonometric functions $\mathbf{C}_g \cdot \cos(\boldsymbol{\varphi}_g)$ and $\mathbf{C}_g \cdot \sin(\boldsymbol{\varphi}_g)$ for real and imaginary components; (4) Plane-specific early termination when transmittance falls below threshold ϵ_T; (5) Efficient shared memory usage for tile-based processing with complex-valued intermediate storage.

D.4 Forward Pass

D.4.1 Complex Field Generation For each pixel (x, y) in each plane Π_l, the complex field U is:

$$U_{\Pi_l}(x, y) = \text{real}_{\Pi_l}(x, y) + i \cdot \text{imag}_{\Pi_l}(x, y) \tag{6}$$

Where for each Gaussian $\mathcal{G}_n$:

$$\begin{aligned} \text{real}_{\Pi_l}(x,y) &= \sum_n \mathbf{c}_n \cdot \boldsymbol{\alpha}_n \cdot T_n \cdot \cos(\boldsymbol{\varphi}_n) \\ \text{imag}_{\Pi_l}(x,y) &= \sum_n \mathbf{c}_n \cdot \boldsymbol{\alpha}_n \cdot T_n \cdot \sin(\boldsymbol{\varphi}_n) \end{aligned} \tag{7}$$

D.4.2 Transmittance Calculation

$$T_n = \prod_{j=1}^{n-1}(1-\boldsymbol{\alpha}_j) \tag{8}$$

D.4.3 Alpha Calculation

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_n^{\text{base}} \cdot \mathcal{G}_n^{\text{proj}} \cdot \boldsymbol{\rho}_{n,l} \tag{9}$$

D.4.4 Gaussian Projection (Exponent Term)

$$\begin{aligned} \mathcal{G}_n^{\text{proj}} = \exp\big(-0.5 \cdot (d_x \cdot (\Sigma'^{-1}_{00} \cdot d_x + \Sigma'^{-1}_{10} \cdot d_y) \\ + d_y \cdot (\Sigma'^{-1}_{01} \cdot d_x + \Sigma'^{-1}_{11} \cdot d_y))\big) \end{aligned} \tag{10}$$

Where $d_x = x - \mu_x$ and $d_y = y - \mu_y$.

D.5 Backward Pass

D.5.1 Gradient Flow Overview For each parameter θ, the gradient follows the chain rule from the final loss $\mathcal{L}$ to the parameter:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial P} \cdot \frac{\partial P}{\partial U_{\Pi_l}} \cdot \frac{\partial U_{\Pi_l}}{\partial \theta} \tag{11}$$

where $\mathcal{L}$ is the loss and P is the hologram output.

D.5.2 Detailed Gradient Derivation for Each Parameter

(1) **Gradient for rotation parameters (related to R_n)**
The gradient with respect to rotation parameters follows the chain rule:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{R}_n} = \frac{\partial \mathcal{L}}{\partial P} \cdot \frac{\partial P}{\partial U_{\Pi_l}} \cdot \frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_n} \cdot \frac{\partial \boldsymbol{\alpha}_n}{\partial \mathcal{G}_n^{\text{proj}}} \cdot \frac{\partial \mathcal{G}_n^{\text{proj}}}{\partial \text{term}} \\ \cdot \frac{\partial \text{term}}{\partial \Sigma'^{-1}} \cdot \frac{\partial \Sigma'^{-1}}{\partial \Sigma'} \cdot \frac{\partial \Sigma'}{\partial \Sigma} \cdot \frac{\partial \Sigma}{\partial \mathbf{R}_n} \cdot \frac{\partial \mathbf{R}_n}{\partial \mathbf{q}} \end{aligned} \tag{12}$$

Key components include:

(a) $\frac{\partial \mathcal{L}}{\partial P}$: Gradient of loss with respect to hologram output, provided by optimization algorithm.
(b) $\frac{\partial P}{\partial U_{\Pi_l}}$: Derivative of hologram with respect to complex field, computed via band-limited propagation.
(c) $\frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_i} = \mathbf{c}_i \cdot T_i \cdot e^{j\cdot\boldsymbol{\varphi}_i} - \sum_{k=i+1}^{N} \mathbf{c}_k \cdot \boldsymbol{\alpha}_k \cdot T_k \cdot \frac{\partial T_k}{\partial \boldsymbol{\alpha}_i} \cdot e^{j\cdot\boldsymbol{\varphi}_k}$ where $\frac{\partial T_k}{\partial \boldsymbol{\alpha}_i} = -T_k/(1-\boldsymbol{\alpha}_i)$ for $i < k$.
(d) $\frac{\partial \boldsymbol{\alpha}_n}{\partial \mathcal{G}_n^{\text{proj}}} = \boldsymbol{\alpha}_n^{\text{base}} \cdot \boldsymbol{\rho}_{n,l}$, from $\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_n^{\text{base}} \cdot \mathcal{G}_n^{\text{proj}} \cdot \boldsymbol{\rho}_{n,l}$.
(e) $\frac{\partial \mathcal{G}_n^{\text{proj}}}{\partial \text{term}} = -0.5 \cdot \mathcal{G}_n^{\text{proj}}$, from $\mathcal{G}_n^{\text{proj}} = \exp(-0.5 \cdot \text{term})$.
(f) $\frac{\partial \text{term}}{\partial \Sigma'^{-1}} = \Delta \cdot \Delta^T$ where $\Delta = (p - \mathbf{x}_n^{\text{proj}})$ is the pixel-mean difference.
(g) $\frac{\partial \Sigma'^{-1}}{\partial \Sigma'} = -\Sigma'^{-1} \otimes \Sigma'^{-1}$, following matrix inverse derivative.
(h) $\frac{\partial \Sigma'_{ij}}{\partial \Sigma_{kl}} = \sum_{m,n} J_{im}\mathbf{R}_{mk}\mathbf{R}_{nl}J_{jn}$, where J is the projection Jacobian and $\mathbf{R}$ the rotation matrix.
(i) $\frac{\partial \Sigma}{\partial \mathbf{R}_{ij}} = \frac{\partial(\mathbf{R}\mathrm{S}\mathrm{S}^\top\mathbf{R}^\top)}{\partial \mathbf{R}_{ij}} = E_{ij}\cdot\mathrm{S}\cdot\mathrm{S}^T\cdot\mathbf{R}^T + \mathbf{R}\cdot\mathrm{S}\cdot\mathrm{S}^T\cdot E_{ji}^T$, where E_{ij} is a standard basis matrix.
(j) $\frac{\partial \mathbf{R}_n}{\partial \mathbf{q}} = \frac{\partial \mathbf{R}}{\partial \mathbf{q}}$, where $\mathbf{q}$ is the quaternion representation of rotation.

(2) **Gradient for $\mathbf{x}_n$ (Gaussian centers)**
The gradient for $\mathbf{x}_n$ passes through multiple paths in the computation graph:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_n} = \frac{\partial \mathcal{L}}{\partial P} \cdot \frac{\partial P}{\partial U_{\Pi_l}} \cdot \frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_n} \cdot \frac{\partial \boldsymbol{\alpha}_n}{\partial \mathcal{G}_n^{\text{proj}}} \cdot \frac{\partial \mathcal{G}_n^{\text{proj}}}{\partial \text{term}} \\ \cdot \frac{\partial \text{term}}{\partial \text{diff}} \cdot \frac{\partial \text{diff}}{\partial \mathbf{x}_n^{\text{proj}}} \cdot \frac{\partial \mathbf{x}_n^{\text{proj}}}{\partial \mathbf{x}_n^{\text{cam}}} \cdot \frac{\partial \mathbf{x}_n^{\text{cam}}}{\partial \mathbf{x}_n} \end{aligned} \tag{13}$$

Key components include:

(a) $\frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_n}$: As described above.
(b) $\frac{\partial \boldsymbol{\alpha}_n}{\partial \mathcal{G}_n^{\text{proj}}} = \boldsymbol{\alpha}_n^{\text{base}} \cdot \boldsymbol{\rho}_{n,l}$
(c) $\frac{\partial \mathcal{G}_n^{\text{proj}}}{\partial \text{term}} = -0.5 \cdot \mathcal{G}_n^{\text{proj}}$
(d) $\frac{\partial \text{term}}{\partial \text{diff}} = \Sigma'^{-1} \cdot \text{diff} + \text{diff} \cdot \Sigma'^{-1}$, where $\text{diff} = (p - \mathbf{x}_n^{\text{proj}})$ is the pixel-mean difference
(e) $\frac{\partial \text{diff}}{\partial \mathbf{x}_n^{\text{proj}}} = -I$ (negative identity matrix)
(f) $\frac{\partial \mathbf{x}_n^{\text{proj}}}{\partial \mathbf{x}_n^{\text{cam}}}$: Projection Jacobian
(g) $\frac{\partial \mathbf{x}_n^{\text{cam}}}{\partial \mathbf{x}_n} = \mathbf{R}$ (camera rotation matrix)

Critically, for pixels (x, y) and Gaussian centers (μ_x, μ_y), we set $d_x = x - \mu_x$ and $d_y = y - \mu_y$. The gradient components are:

$$\begin{aligned} \frac{\partial \text{term}}{\partial \mu_x} &= -(-d_x \cdot \text{inv}_{00} - d_y \cdot \text{inv}_{10}) \\ \frac{\partial \text{term}}{\partial \mu_y} &= -(-d_x \cdot \text{inv}_{01} - d_y \cdot \text{inv}_{11}) \end{aligned} \tag{14}$$

Note the negative sign in the implementation that results from $\frac{\partial(x-\mu_x)}{\partial \mu_x} = -1$. This is correctly captured in our final gradient computation.

(3) **Gradient for covariance matrix elements**
The covariance gradient calculation is performed using the chain rule and the matrix inverse derivative formula:

$$\frac{\partial \mathcal{L}}{\partial \Sigma} = -\Sigma^{-1} \cdot \frac{\partial \mathcal{L}}{\partial \Sigma^{-1}} \cdot \Sigma^{-1} \tag{15}$$

For the 2D covariance matrix elements, we compute:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \Sigma_{00}} &= -(bt_{00} \cdot tmp_{00} + bt_{01} \cdot tmp_{10}) \\ \frac{\partial \mathcal{L}}{\partial \Sigma_{01}} &= -(bt_{00} \cdot tmp_{01} + bt_{01} \cdot tmp_{11}) \\ \frac{\partial \mathcal{L}}{\partial \Sigma_{10}} &= -(bt_{10} \cdot tmp_{00} + bt_{11} \cdot tmp_{10}) \\ \frac{\partial \mathcal{L}}{\partial \Sigma_{11}} &= -(bt_{10} \cdot tmp_{01} + bt_{11} \cdot tmp_{11}) \end{aligned} \tag{16}$$

where:

- bt is the transpose of the inverse covariance matrix
- tmp represents intermediate gradient values from the quadratic form

This ensures that the negative sign from the matrix inverse derivative is properly incorporated.

(4) **Gradient for S_n (Gaussian scales)**

$$\frac{\partial \mathcal{L}}{\partial \mathbf{S}_n} = \frac{\partial \mathcal{L}}{\partial P} \cdot \frac{\partial P}{\partial U_{\Pi_l}} \cdot \frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_n} \cdot \frac{\partial \boldsymbol{\alpha}_n}{\partial \mathcal{G}_n^{\text{proj}}} \cdot \frac{\partial \mathcal{G}_n^{\text{proj}}}{\partial \text{term}} \cdot \frac{\partial \text{term}}{\partial \Sigma'^{-1}} \cdot \frac{\partial \Sigma'^{-1}}{\partial \Sigma'} \cdot \frac{\partial \Sigma'}{\partial \Sigma} \cdot \frac{\partial \Sigma}{\partial \mathbf{S}_n} \tag{17}$$

(a) Components from $\frac{\partial \mathcal{L}}{\partial P}$ through $\frac{\partial \Sigma'}{\partial \Sigma}$ as previously described.
(b) $\frac{\partial \Sigma_{ij}}{\partial \mathbf{S}_{n,k}} = 2 \cdot \mathbf{S}_{n,k} \cdot \mathbf{R}_{i,k} \cdot \mathbf{R}_{j,k}$ for anisotropic case.

(5) **Gradient for $\mathbf{c}_n$ (Gaussian amplitudes)**

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}_n} = \frac{\partial \mathcal{L}}{\partial P} \cdot \frac{\partial P}{\partial U_{\Pi_l}} \cdot \frac{\partial U_{\Pi_l}}{\partial \mathbf{c}_n} \tag{18}$$

(a) $\frac{\partial U_{\Pi_l}}{\partial \mathbf{c}_n} = \boldsymbol{\alpha}_n \cdot T_n \cdot e^{j \cdot \boldsymbol{\varphi}_n}$, directly contributing to the complex field.

In the implementation, this becomes:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}_n} = \alpha_n \cdot T_{\text{before}} \cdot (\cos(\varphi_n) \cdot \frac{\partial \mathcal{L}}{\partial \text{real}} + \sin(\varphi_n) \cdot \frac{\partial \mathcal{L}}{\partial \text{imag}}) \tag{19}$$

(6) **Gradient for $\boldsymbol{\varphi}_n$ (Gaussian phases)**

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varphi}_n} = \frac{\partial \mathcal{L}}{\partial P} \cdot \frac{\partial P}{\partial U_{\Pi_l}} \cdot \frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\varphi}_n} \tag{20}$$

(a) $\frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\varphi}_n} = j \cdot \mathbf{c}_n \cdot \boldsymbol{\alpha}_n \cdot T_n \cdot e^{j \cdot \boldsymbol{\varphi}_n}$, from the derivative of the complex exponential.

This expands to:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varphi}_n} = \alpha_n \cdot T_{\text{before}} \cdot \mathbf{c}_n \cdot (-\sin(\varphi_n) \cdot \frac{\partial \mathcal{L}}{\partial \text{real}} + \cos(\varphi_n) \cdot \frac{\partial \mathcal{L}}{\partial \text{imag}}) \tag{21}$$

Note the negative sign on the sine term, which comes from the derivative of $\cos(\varphi_n)$.

(7) **Gradient for $\boldsymbol{\alpha}_n^{\text{base}}$ (Gaussian base opacities)**

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_n^{\text{base}}} = \frac{\partial \mathcal{L}}{\partial P} \cdot \frac{\partial P}{\partial U_{\Pi_l}} \cdot \frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_n} \cdot \frac{\partial \boldsymbol{\alpha}_n}{\partial \boldsymbol{\alpha}_n^{\text{base}}} \tag{22}$$

(a) $\frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_n}$: As previously described.
(b) $\frac{\partial \boldsymbol{\alpha}_n}{\partial \boldsymbol{\alpha}_n^{\text{base}}} = \mathcal{G}_n^{\text{proj}} \cdot \boldsymbol{\rho}_{n,l}$ from $\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_n^{\text{base}} \cdot \mathcal{G}_n^{\text{proj}} \cdot \boldsymbol{\rho}_{n,l}$.

In the implementation, this is calculated as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_n^{\text{base}}} = \exp(\text{power}) \cdot \boldsymbol{\rho}_{n,l} \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_n} \tag{23}$$

(8) **Gradient for $\boldsymbol{\rho}_{n,l}$ (plane assignment probabilities)**

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\rho}_{n,l}} = \frac{\partial \mathcal{L}}{\partial P} \cdot \frac{\partial P}{\partial U_{\Pi_l}} \cdot \frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_n} \cdot \frac{\partial \boldsymbol{\alpha}_n}{\partial \boldsymbol{\rho}_{n,l}} \tag{24}$$

(a) $\frac{\partial U_{\Pi_l}}{\partial \boldsymbol{\alpha}_n}$: As previously described.
(b) $\frac{\partial \boldsymbol{\alpha}_n}{\partial \boldsymbol{\rho}_{n,l}} = \boldsymbol{\alpha}_n^{\text{base}} \cdot \mathcal{G}_n^{\text{proj}}$

In the implementation, the gradient for plane probabilities is:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\rho}_{n,l}} = \boldsymbol{\alpha}_n^{\text{base}} \cdot \exp(\text{power}) \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_n} \tag{25}$$

D.5.3 Multi-Plane Transmittance Calculation For multiple planes, the transmittance is calculated separately for each plane l:

$$T_n^{(l)} = \prod_{j=1}^{n-1} (1 - \boldsymbol{\alpha}_j^{(l)}), \quad \text{where} \quad \boldsymbol{\alpha}_j^{(l)} = \boldsymbol{\alpha}_j^{\text{base}} \cdot \mathcal{G}_j^{\text{proj}} \cdot \boldsymbol{\rho}_{j,l} \tag{26}$$

The gradient of transmittance for plane l is:

$$\frac{\partial T_k^{(l)}}{\partial \boldsymbol{\alpha}_n^{(l)}} = \begin{cases} -T_k^{(l)} / (1 - \boldsymbol{\alpha}_n^{(l)}) & \text{if } n < k \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

This approach ensures each plane's rendering is treated independently, with plane weights determining which planes a Gaussian contributes to. The CUDA implementation computes gradients per-pixel and per-Gaussian for each plane, with gradients accumulated atomically across all planes.

E Ablation Study of CUDA operations

This section conduct ablation study on the contribution of our two key method components: tile-based complex-valued rasterization and FFT-based layer propagation. We measure performance using *torch.profiler*, reporting GPU time. Each iteration is wrapped by explicit profiling scopes, and the two components are isolated as timing windows; all CUDA kernels executed within each window are collected via hierarchical traversal and accumulated using self-device time to avoid double-counting. Per-kernel GPU times are averaged over 50 iterations to obtain a stable per-step runtime breakdown, without inserting any explicit CUDA synchronization that would perturb normal execution.

Fig. 3 presents a detailed profiling analysis of the computational breakdown for each rendering step. Averaged by 50 steps, the complex-valued splatting and forward recording step completes in 10.11 ms, with tile-based splatting (SplatTileCuda) accounting for 5.18 ms (51.2%), making it the dominant operation. Additional costs include batch matrix operations (1.71 ms, 16.9%) and FFT operations for band-limited propagation (1.44 ms, 14.3%).

The inverse propagation step completes in 9.00 ms, dominated by FFT-based complex-to-complex transformations (4.43 ms, 49.2%) for multi-plane propagation. Matrix multiplications contribute 2.27 ms (25.3%), while tensor roll operations for FFT alignment require 1.87 ms (20.7%). Together, tile-based rasterization and FFT-based propagation constitute 9.61 ms of the total 19.11 ms pipeline (50.3%), demonstrating that these two techniques are the primary computational contributors enabling real-time performance. The remaining overhead from tensor operations, sorting, and memory management represents 9.50 ms (49.7%).

F Gaussian Distribution And Phase Characteristics Analysis

This section presents a quantitative analysis of the learned complex-valued holographic radiance field, focusing on phase behavior, spatial frequency content, and their relationship to Gaussian primitive properties.

F.1 Phase and Amplitude Distributions

The learned hologram phase exhibits a structured distribution that lies between globally smooth and fully random patterns, with a

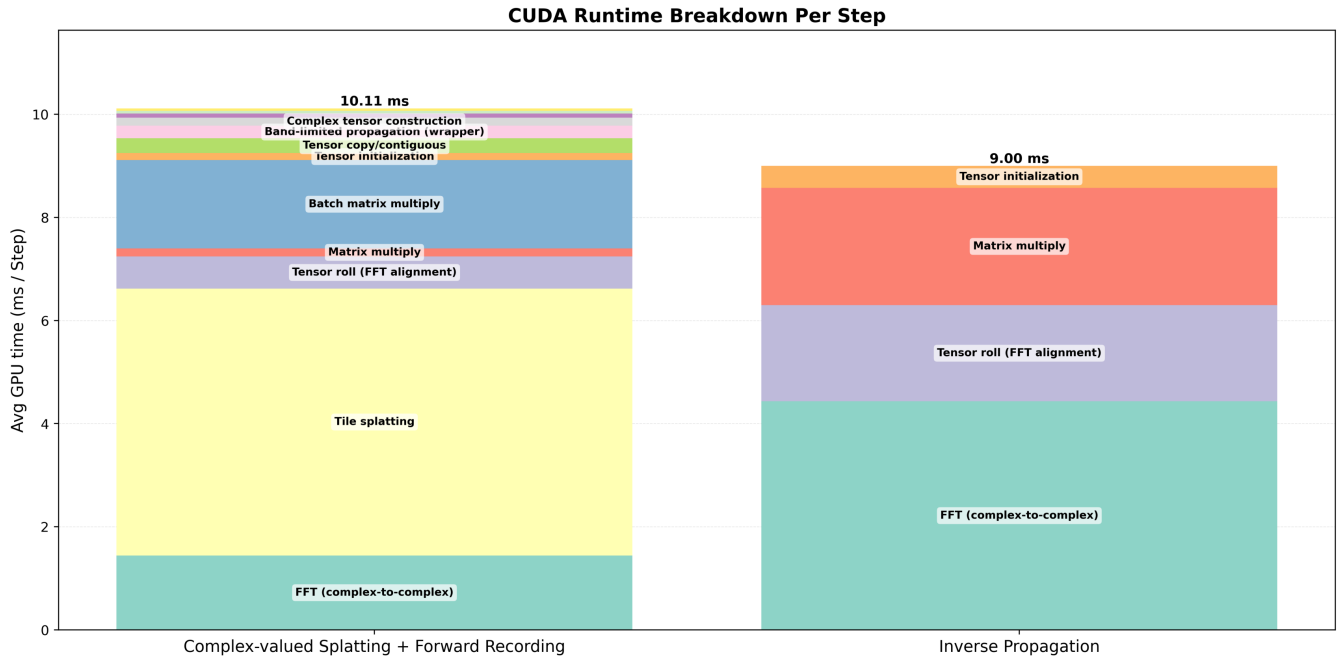

Fig. 3. Ablation study of CUDA runtime breakdown per step.

clear bias toward spatial smoothness. As shown in Fig. 4(j), the phase histogram is bimodal with peaks near 0 and 2π, indicating non-uniform but organized phase statistics.

This behavior arises from the coherent superposition of multiple complex-valued Gaussian primitives. Each Gaussian acts as a spatially localized, low-pass basis function, enforcing continuous phase variation within its finite support. Consequently, individual primitives cannot generate high-frequency, pixel-wise phase fluctuations.

The final phase field is formed by summing many such locally smooth contributions, each associated with an independently learned phase parameter. Their interference introduces structured, higher-frequency variations while preserving local continuity, yielding phase patterns that are irregular yet not fully random.

Two factors jointly govern this behavior: (1) the Gaussian primitives impose local smoothness due to their finite spatial support; (2) phase parameters $\boldsymbol{\varphi}_n$ are learned per primitive rather than per pixel, inducing spatially correlated phase variations. As illustrated in Fig. 4(b), the resulting phase map exhibits extended coherent regions with sharper transitions primarily at overlap regions and object boundaries.

The amplitude distribution Fig. 4(k) is strongly right-skewed, with most values near zero and a long tail at higher amplitudes. This reflects the sparsity of the representation, where only a subset of Gaussians contribute significantly at each spatial location. Low amplitudes correspond to background regions, while the tail captures areas of constructive interference. The amplitude map Fig. 4(a) confirms that energy is concentrated in object regions.

F.2 Spatial Coherence and Gradient Analysis

The low-pass nature of Gaussian primitives induces spatially smooth phase structure, as shown by the phase gradient magnitude in Fig. 4(e). Gradients remain low across most of the hologram plane, increasing primarily at object boundaries and depth discontinuities. This behavior follows directly from the continuous Gaussian projections.

Interference between overlapping Gaussians introduces localized high-frequency variations. The amplitude gradient map (Fig. 4(d)) shows sharp transitions concentrated at object edges, where multiple primitives with differing phases overlap.

F.3 Spatial Frequency Characteristics

Fourier-domain analysis Fig. 4(c,f) reveals a spectrum dominated by low-frequency components, consistent with smooth spatial variations from Gaussian primitives. Nevertheless, substantial mid-to-high frequency energy is present near the center, encoding fine interference details required for accurate holographic reconstruction. The spatial frequency profile decays monotonically from DC.

F.4 Gaussian Primitive Characteristics

The learned Gaussian scale distribution Fig. 4(g) is heavily concentrated in the range 0.00–0.05, with exponential decay toward larger scales. This reflects the densification strategy's preference for small primitives in geometrically complex regions, enabling high-resolution modeling while maintaining efficiency, this is consistent with traditional 3DGS. The dominance of small-scale Gaussians further promotes local phase coherence due to their limited spatial support.

The opacity distribution Fig. 4(h) is bimodal, with peaks near 0.0 and 1.0, indicating specialization toward either transparent or opaque states. The mean opacity near 0.5 reflects a balance between coverage and transmittance, enabling effective modeling of wavefront occlusion via accumulated transmittance.

F.5 Depth Distribution and Plane Assignment

The depth distribution Fig. 4(i) shows Gaussians clustering around predefined depth planes, confirming effective learning of plane assignment probabilities. This validates the effectiveness of the STE-based plane assignment.

For multi-plane settings, the plane assignment statistics (Fig. 4(l)) show balanced utilization across planes (e.g., 51.4% vs. 48.6%), indicating no collapse to a single plane. As shown in the first row's zoomed insets of Fig. 5, the high-frequency details of Gaussians for front and back focus are gathered at separate depth planes, demonstrating the effectiveness of the plane splitting of our method.

F.6 Relationship to Holographic Reconstruction

Fig. 5 demonstrates how the learned parameters in our method will affect reconstruction. Multi-plane amplitude and phase maps exhibit coherent phase structure within objects and localized discontinuities at depth boundaries, consistent with the statistical analysis.

During forward recording, plane-specific complex fields are aggregated via band-limited ASM propagation (H_{Z_l}). Inverse propagation (H_{-Z_l}) produces reconstructed intensities with natural defocus blur. In-focus regions remain sharp, while out-of-focus areas exhibit smooth diffraction blur without structured artifacts existed in the existing learned CGH methods.

F.7 Novel View Synthesis Progression

Fig. 6 demonstrates the behavior of learned Gaussian primitives under continuous camera trajectory variation, validating our implicit geometric learning mechanism.

The first two rows show Plane 1 and Plane 2 complex-field projections (amplitude and phase). Despite using fixed plane assignment probabilities $\boldsymbol{\rho}$ learned during training, the projected complex fields maintain correct geometric appearance across novel views. This occurs because:

(1) **Occlusion handling through alpha blending**: Gaussians that become occluded at novel angles are naturally attenuated by the transmittance product $\prod_{j=1}^{n-1}(1-\alpha_j)$.
(2) **Multi-view learned assignments**: The plane probabilities $\boldsymbol{\rho}_n$ are optimized jointly over the full training viewpoint distribution, enabling them to function collectively rather than being viewpoint-specific.
(3) **Geometric awareness**: While no Gaussian is explicitly supervised with a ground-truth 3D depth during forward recording, their collective behavior— through learned $\boldsymbol{\rho}$ and α parameters—produces geometrically consistent results, which empirically generalize well to continuously interpolated novel views.

To further clarify this behavior, consider a single Gaussian primitive in our representation. Each Gaussian is parameterized by both a plane assignment probability $\boldsymbol{\rho}$ and an opacity α (Eq. 9). These parameters are learned through multi-view reconstruction supervision and are fixed after training, making the Gaussian intrinsically view-independent and directly reusable for arbitrary novel viewpoints. While prior methods do not introduce an explicit plane assignment parameter $\boldsymbol{\rho}$, the role of α in our formulation is consistent with recent work [Choi et al. 2025], where opacity similarly governs occlusion and contribution strength.

The final column (Novel View 120) demonstrates robustness under large rotation angles, where the camera viewpoint transitions from one side of the chair to the opposite side. At this extreme angle, many Gaussians that contributed significantly at earlier views become heavily occluded; nevertheless, the reconstruction remains correct because other Gaussians that were previously attenuated now contribute more strongly. The zoomed insets reveal that individual Gaussians maintain consistent intrinsic properties (amplitude and phase) across viewpoints, while their effective contributions vary naturally due to occlusion. This validates that our method achieves view-consistent holographic encoding through learned scene parameters rather than per-view recalculation. The forward-recorded hologram (Row 3) and reconstructed intensities (Row 4) further confirm that fixed plane assignments, combined with proper occlusion modeling, enable correct rendering throughout the camera trajectory.

G Extra Simulated And Captured Result

This section presents additional experimental results demonstrating the effectiveness of our complex-valued 3D Gaussian splatting approach across diverse scenes. We provide comprehensive comparisons on both synthetic and real-world datasets, including scenes from the NeRF Synthetic dataset (lego in Fig. 7, chair in Fig. 8, ship in Fig. 9, materials in Fig. 10, and hotdog in Fig. 11) and the LLFF dataset (flower in Fig. 12, fern in Fig. 13, trex in Fig. 14, and room in Fig. 15). Each figure showcases the complete pipeline from radiance field projections and complex 3D hologram rendering to both simulated and experimentally captured results, demonstrating consistent quality across different scene types and lighting conditions.

References

Suyeon Choi, Brian Chao, Jacqueline Yang, Manu Gopakumar, and Gordon Wetzstein. 2025. Gaussian Wave Splatting for Computer-Generated Holography. *arXiv preprint arXiv:2505.06582* (2025).

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

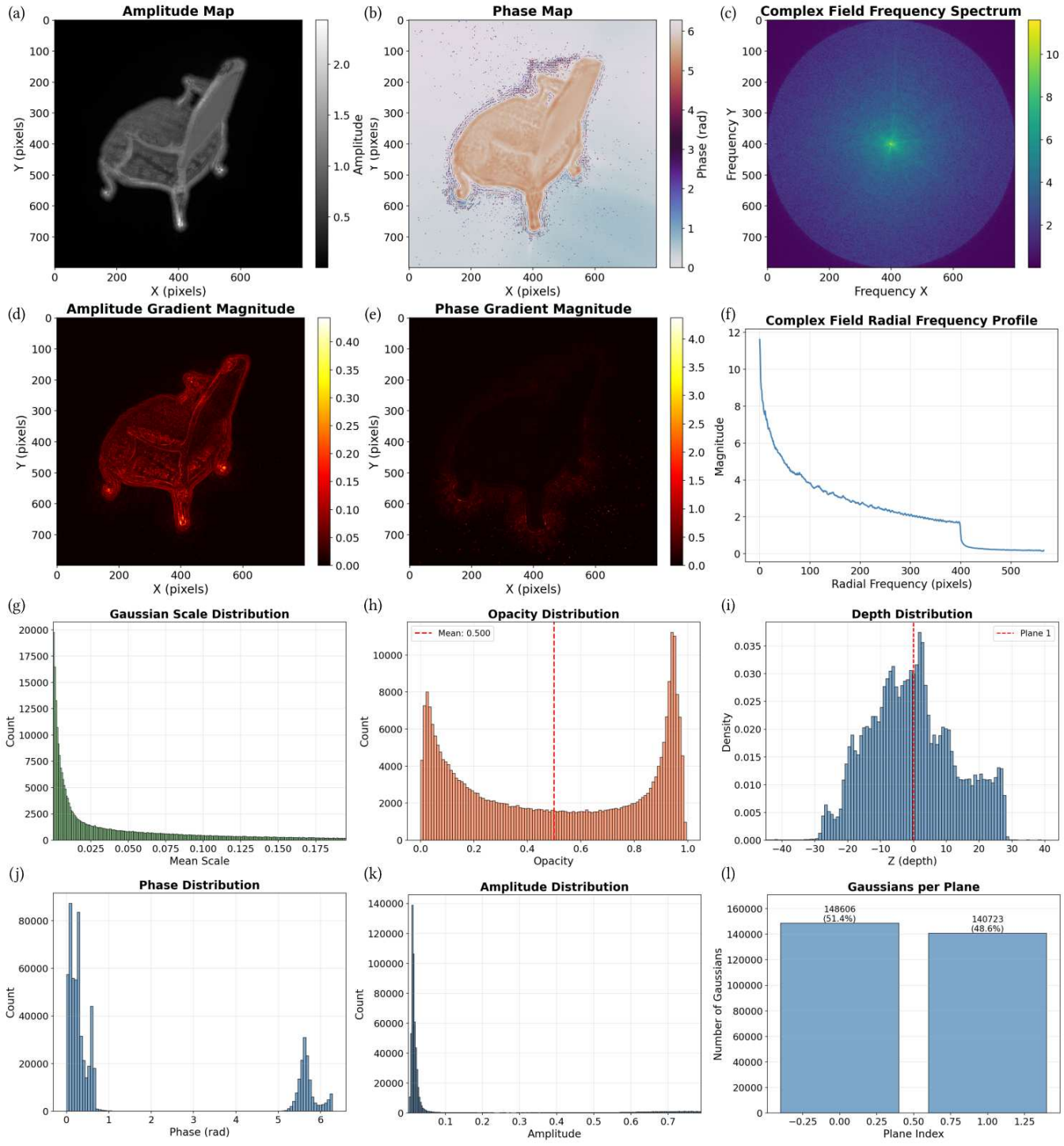

Fig. 4. Statistical analysis of learned complex-valued holographic radiance field properties. **(a)** Amplitude map. **(b)** Phase map. **(c)** Fourier-domain energy distribution. **(d)** Amplitude gradient magnitude. **(e)** Phase gradient magnitude. **(f)** Spatial frequency profile. **(g)** Gaussian scale distribution. **(h)** Opacity distribution. **(i)** Depth distribution. **(j)** Phase distribution. **(k)** Amplitude distribution. **(l)** Gaussian assignment across two depth planes.

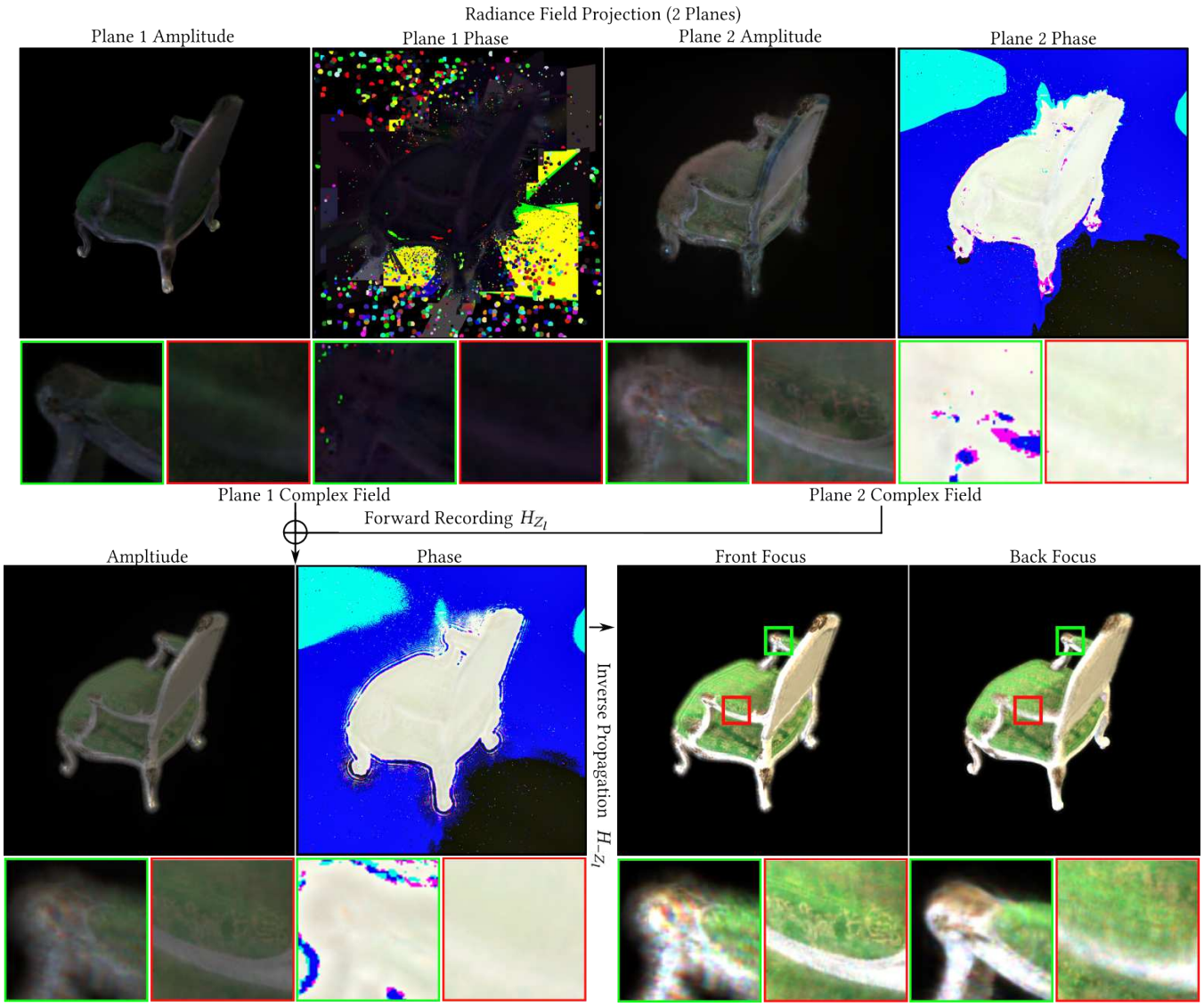

Fig. 5. Visualization of complex-valued holographic radiance field rendering pipeline and holographic reconstruction. **Top:** Multi-plane radiance field projections showing amplitude and phase maps for 2 Depth Planes. **Bottom Left** Forward recording process combines plane-specific complex fields via band-limited ASM (H_{Z_l}) to generate the final hologram. **Bottom Right:** Reconstructed Intensities for front and back focus via Inverse Propagation (H_{-Z_l}).

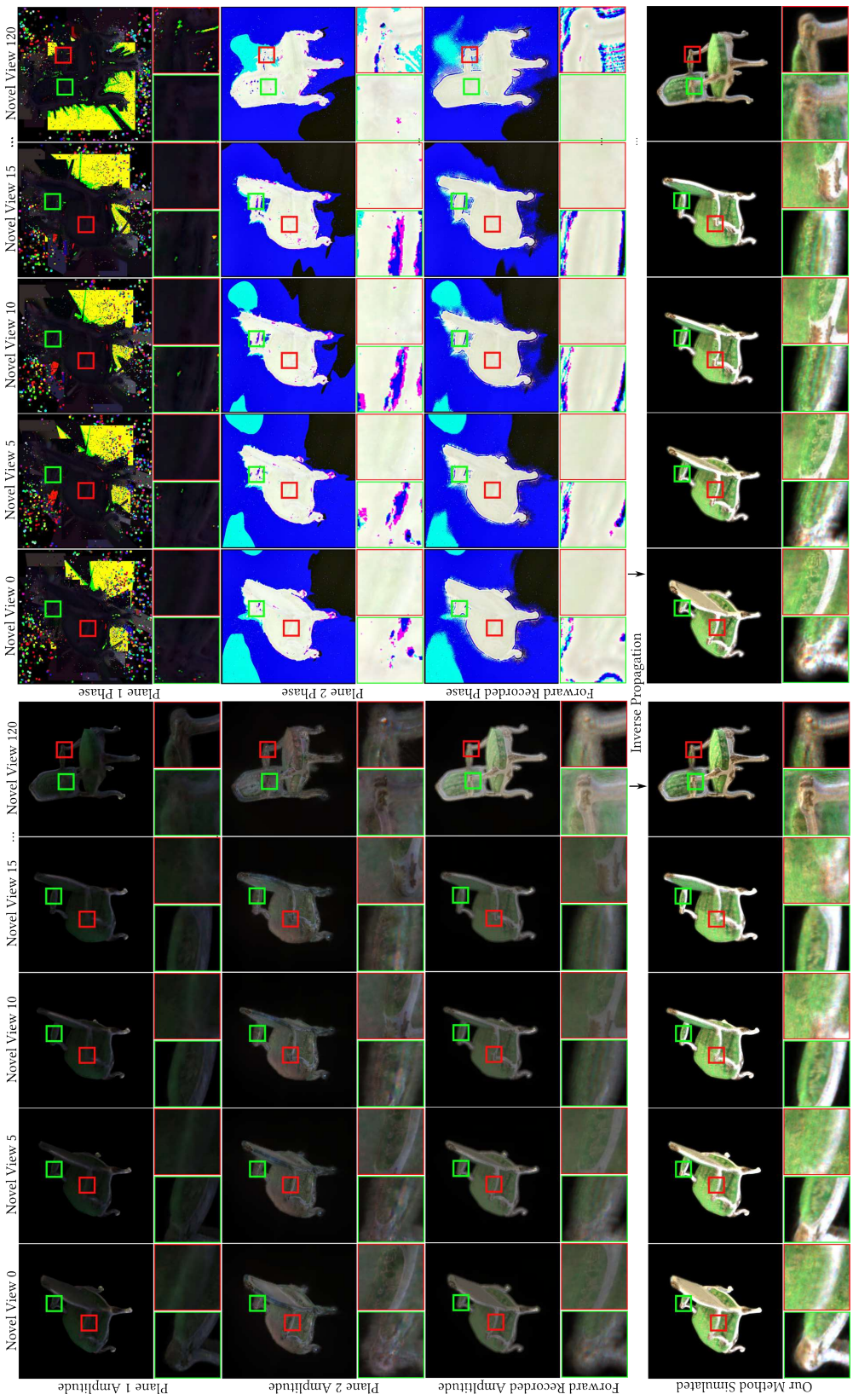

Fig. 6. Gaussian primitive contributions under continuous novel-view variation. Columns show novel viewpoints sampled along a continuous camera trajectory. **Row 1:** Plane 1 complex-field projections from Trained 3D complex-valued Gaussians, visualized as amplitude (left) and phase (right). **Row 2:** Plane 2 complex-field projections. **Row 3:** Forward-recorded hologram formed by coherently combining both depth-plane fields via band-limited ASM propagation. **Row 4:** Reconstructed intensities via inverse propagation, showing front- and back-focus responses.

Fig. 7. Extra novel-view comparison of our method on lego scene from the NeRF Synthetic dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.

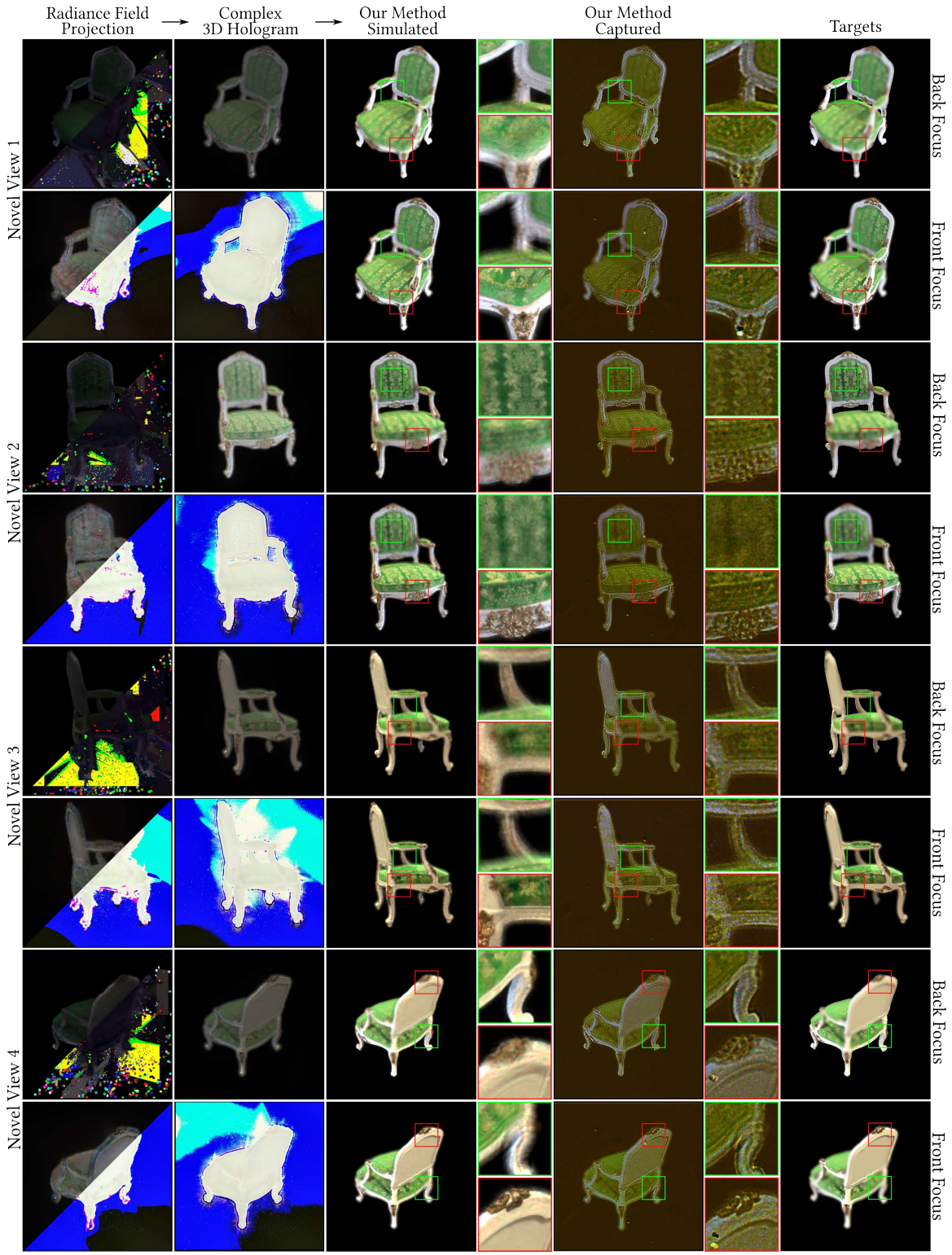

Fig. 8. Extra novel-view comparison of our method on chair scene from the NeRF Synthetic dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.

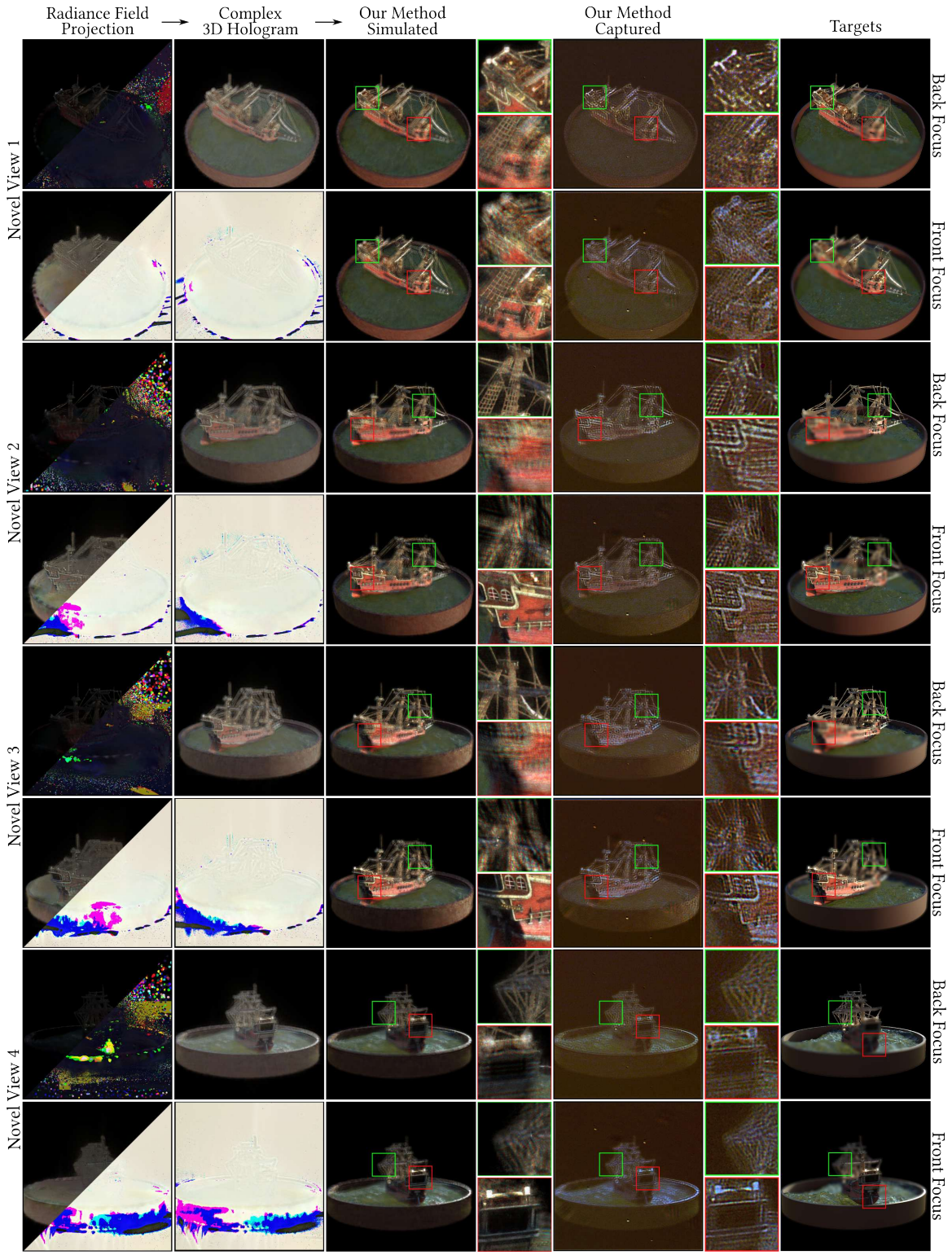

Fig. 9. Extra novel-view comparison of our method on ship scene from the NeRF Synthetic dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.

Fig. 10. Extra novel-view comparison of our method on materials scene from the NeRF Synthetic dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.

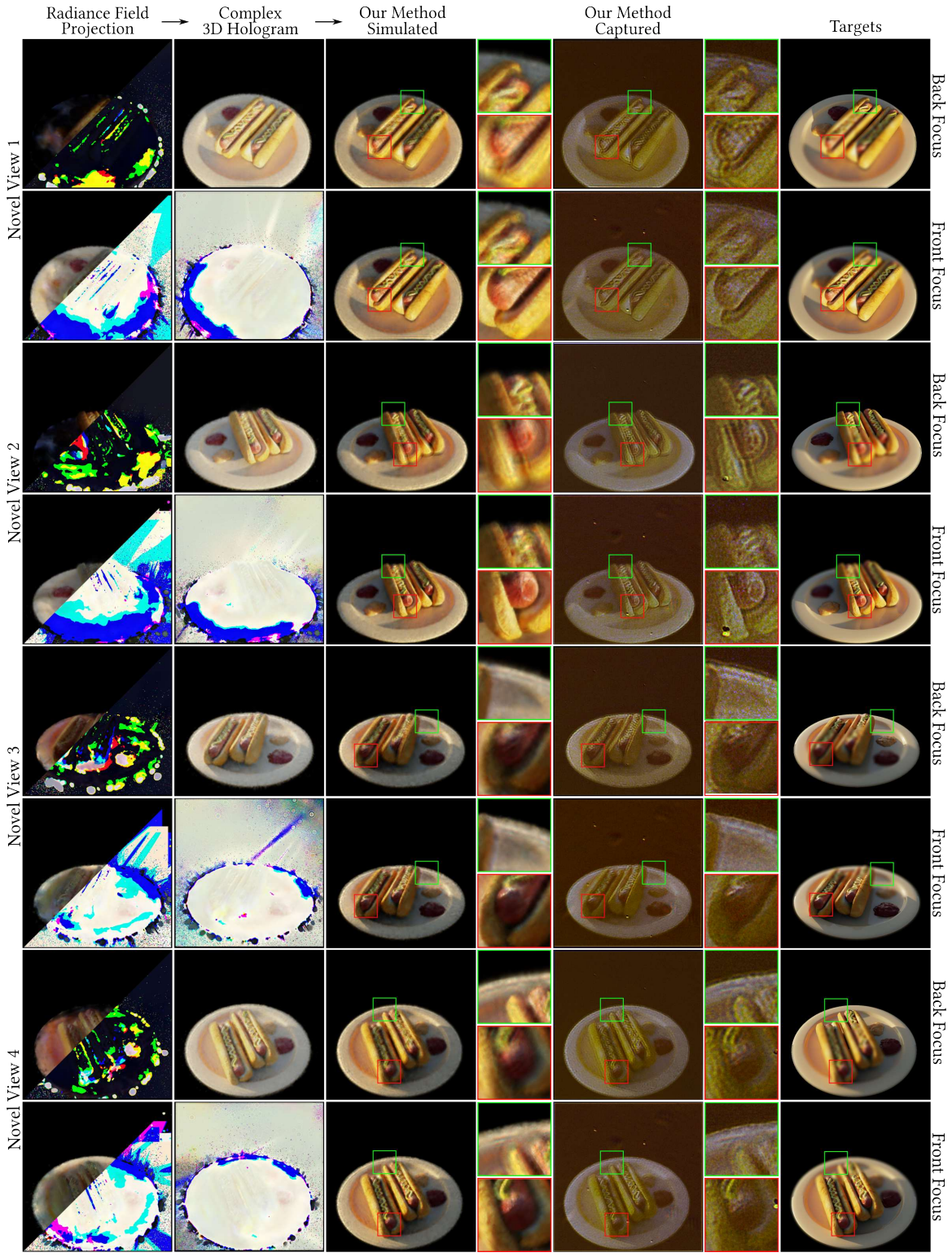

Fig. 11. Extra novel-view comparison of our method on hotdog scene from the NeRF Synthetic dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.

Fig. 12. Extra novel-view comparison of our method on flower scene from the LLFF dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.

Fig. 13. Extra novel-view comparison of our method on fern scene from the LLFF dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.

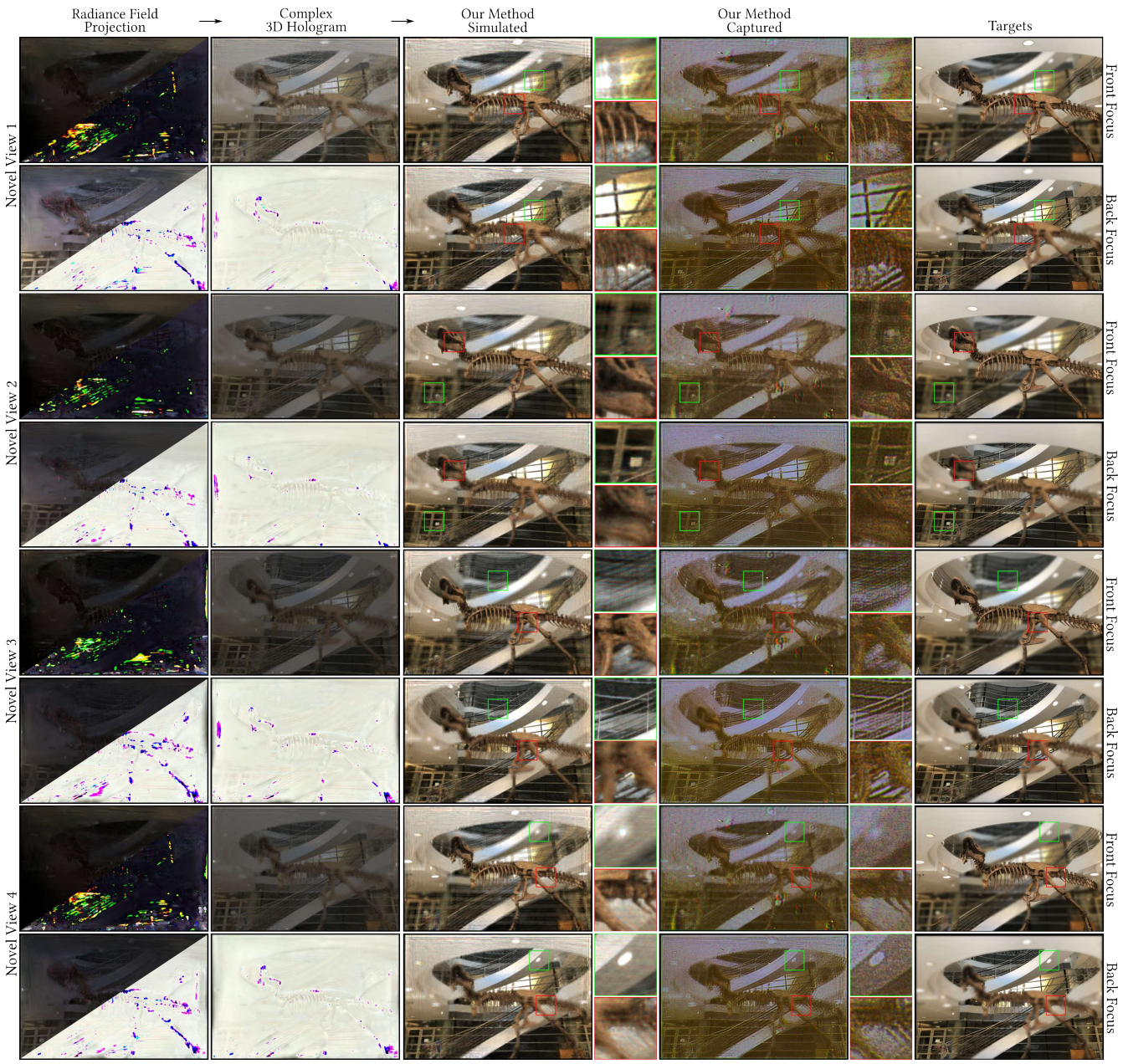

Fig. 14. Extra novel-view comparison of our method on trex scene from the LLFF dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.

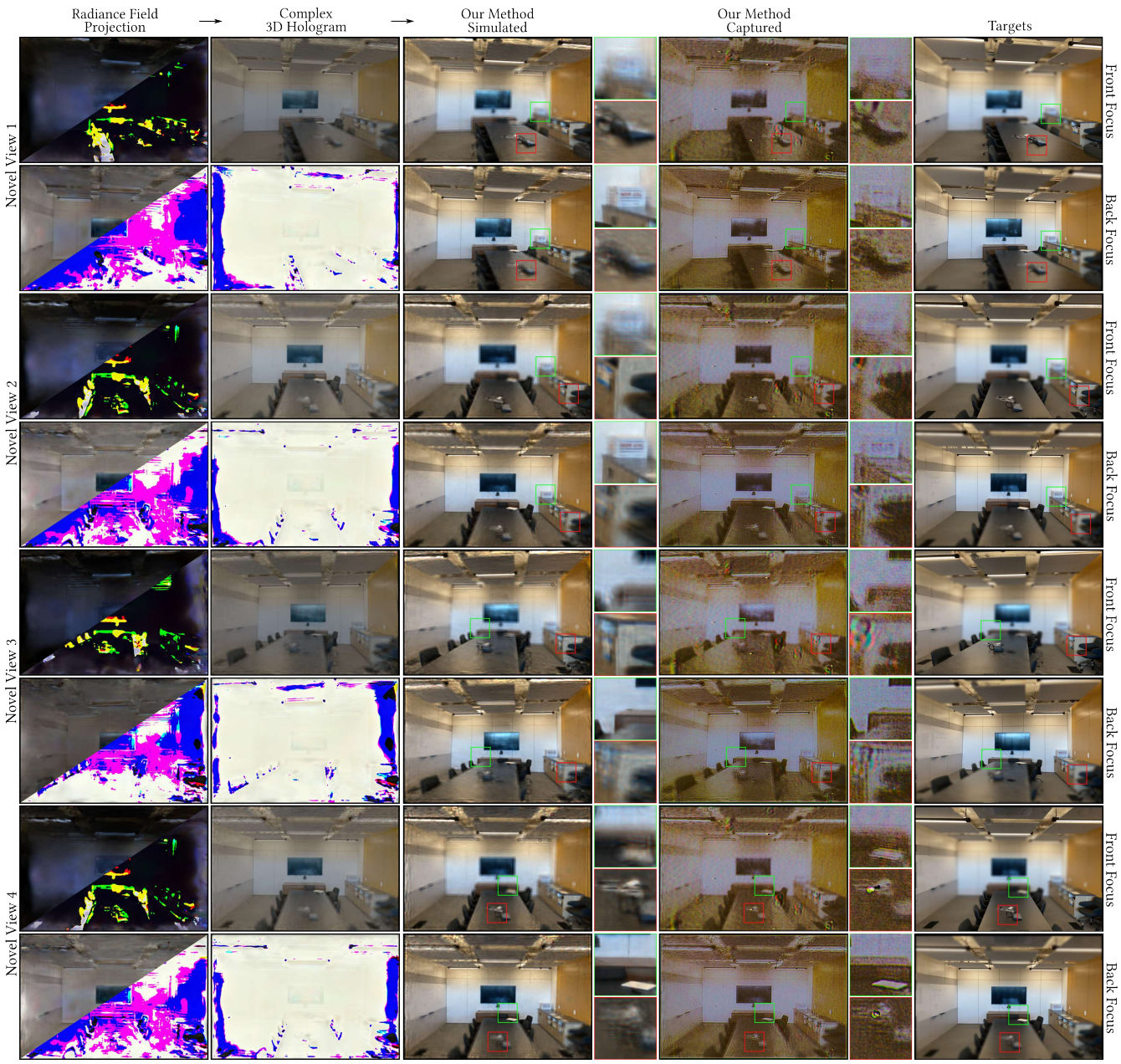

Fig. 15. Extra novel-view comparison of our method on room scene from the LLFF dataset. The first two columns show the radiance field projections and their rendered complex 3D holograms. The central columns present our method's simulated results and experimentally captured results. The rightmost column displays the target images used as optimization objectives in our method.