8. Supplementary Material

8.1. Training Setup

In all our experiments, we train the MLPs in PyTorch using the Adam optimizer with default parameters. We supervise the network based on the mean squared error loss between reconstructed color and ground truth color. We adopt the cosine annealing learning rate scheduler, with initial learning rate as 1×10^{-5} and final learning rate as 1×10^{-8}. We randomly shuffled all the training samples and divided them into batches of size equal to 2048. To ensure reproducibility, we set the random seed as 0.

Unless otherwise noted, we set all networks to have 10 layers with residual connections, where the intermediate layers having a dimension size of 512×512. We apply layer normalization after all MLP layers except the final one.

For networks with sine activation functions, we follow the initialization method suggested by Sitzmann *et al*. [38].

8.2. Additional Tables and Figures

Table 3: **Details of the light field scenes used in experiments.** I_x and I_y refer to width and height of a single image, I_u and I_v refer to angular width and height for the different viewpoints, I_t refers to the number of time steps recorded in a video, and raw byte sizes are calculated with each RGB pixel represented by three bytes (one byte per channel).

Scene	I_u	I_v	I_x	I_y	I_t	Raw Size (MB)
Lego	17	17	1024	1024	1	909
Tarot	17	17	1024	1024	1	909
Bracelet	17	17	1024	640	1	568
Painter	4	4	2048	1088	50	5252
Trains	4	4	2048	1088	50	5252

Table 4: **Network parameters used to achieve results shown in Table 1.** $C = C_u + C_v + C_x + C_y + C_t$ as defined in Section 4. M is the dimension of the MLP layers. For simplicity we use $M \times M$ square matrices for all intermediate layers, where $M = 512$. L refers to the number of MLP layers, including the first and the last layer.

Scene	L	C	C_u	C_v	C_x	C_y	C_t
Lego	10	512	16	16	240	240	0
Tarot	10	512	16	16	240	240	0
Bracelet	10	2048	144	144	1080	680	0
Painter	10	512	0	0	260	180	72
Trains	10	512	0	0	260	180	72

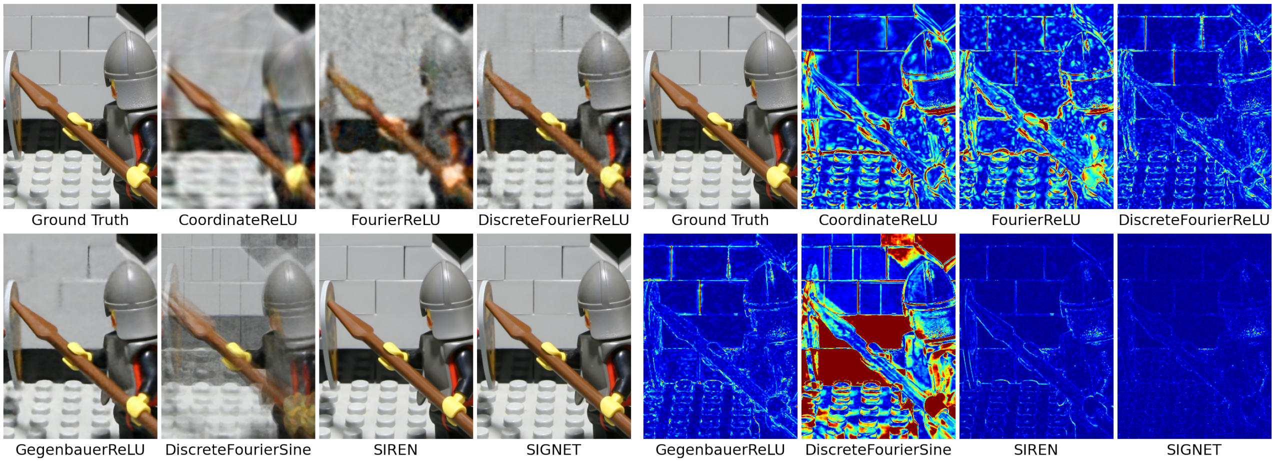

Figure 10: **Examples of reconstruction results (left) and absolute errors (right) on the *Lego* scene.** Our SIGNET method with Gegenbauer input transformation produces the best result. Notice the *SIREN* method is relatively inaccurate in capturing the discontinuity near the edges.

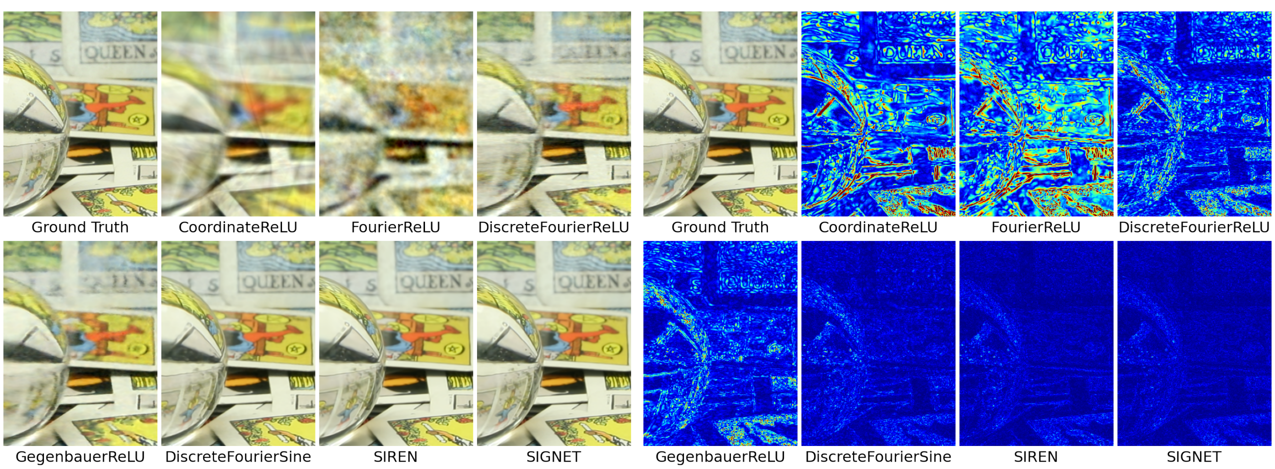

Figure 11: **Examples of reconstruction results (left) and absolute errors (right) on the *Tarot* scene.** The last three methods (DiscreteFourierSine, SIREN, and SIGNET) achieve similarly accurate reconstruction, with most of the residual errors coming from the extreme refraction at the surface of the crystal ball.

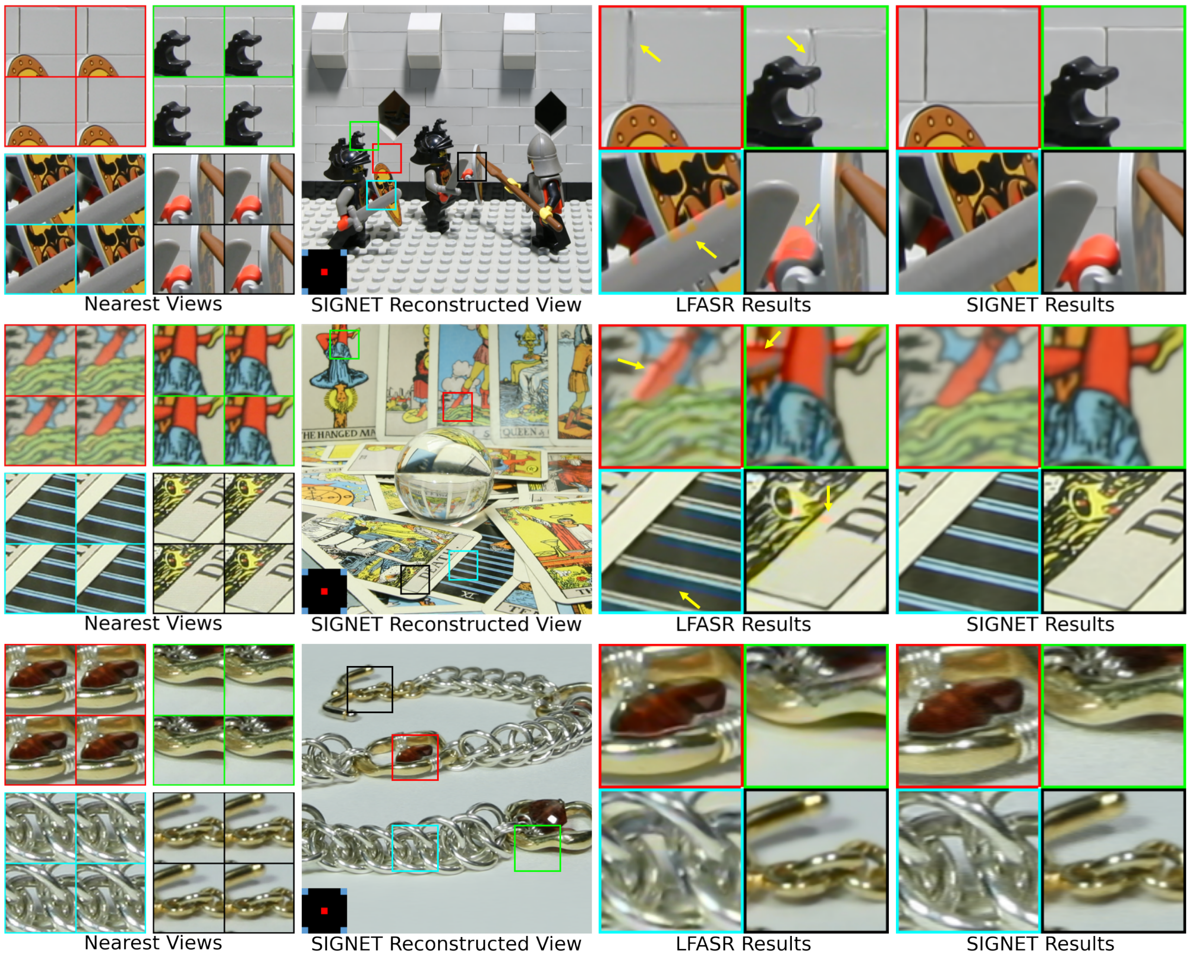

Figure 12: **Angular Upsampling.** At the bottom left corner of the reconstructed view, we show the relative positions of the reconstructed view (red square) and its four nearest views (blue squares) in the original light field. We present reconstructions at novel viewpoints from the three static scenes, and we also show results from the deep-learning-based method, *LFASR* [19], which is trained specifically for light field angular upsampling. Notice in the first two scenes, the LFASR results show visible artifacts pointed to by the yellow arrows, such as distorted geometry and ghosting.