This section provides additional quantitative results and implementation details for LightCtrl omitted from the main paper for brevity.

A. Additional Quantitative Results

Table 1. Scene-level relighting comparison on the **MIIW** testset. **Bold** indicates the best performance.

Method	RMSE ↓	SSIM ↑	PSNR ↑
RGB↔X [3]	0.389	0.425	8.25
IC-Light [4]	0.413	0.337	7.94
LumiNet [2]	**0.139**	**0.904**	17.20
Ours	0.167	0.655	**18.30**

Despite being trained primarily on synthetic object-level data, LightCtrl achieves the highest PSNR (18.30) on the MIIW scene-level benchmark, demonstrating strong generalization to real indoor environments with complex geometry and mixed materials.

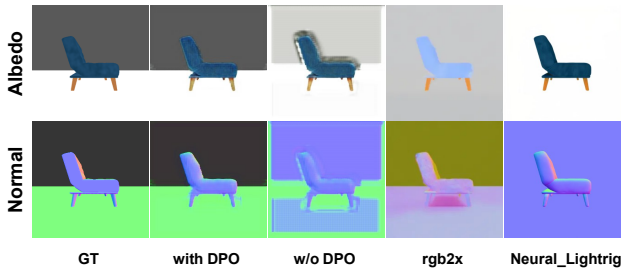

Figure 1. Visual comparison of intrinsic decomposition on a chair object. From left to right: ground-truth (GT), our method with DPO, our method without DPO, rgb2x [3], and Neural_Lightrig [1]. DPO fine-tuning clearly suppresses artifacts and produces more accurate albedo and normal predictions.

B. Implementation Details

We first describe how the relative illumination encoding $\Delta\ell$ is constructed from physically meaningful lighting parameters, with emphasis on the conversion from user-specified yaw–pitch edits to the spherical-harmonic (SH) representation used by the model. We then elaborate on the design of the lighting-aware mask and its conversion into a spatial weighting scheme for denoising. Finally, we detail how appearance, lighting, and proxy tokens are integrated into the conditioning architecture to achieve controllable relighting.

C. Relative Illumination Encoding $\Delta\ell$

Our illumination encoding is designed to support intuitive user control while remaining stable for learning. In practice, lighting edits are specified in a 2D *yaw–pitch* space—a natural interface for rotating a directional light around an object. However, diffusion models operate on vector-valued embeddings, not angles. We therefore convert yaw–pitch edits into a physically meaningful representation through the following steps:

Step 1: From yaw–pitch control to 3D light direction. Given a source lighting configuration with angular parameters (θ_s, ϕ_s) and a user-specified edit defining a target illumination (θ_t, ϕ_t), we map each pair into a unit direction vector in the camera coordinate frame:

$$\boldsymbol{\omega}(\theta, \phi) = \begin{bmatrix} \cos\theta \, \sin\phi \\ \sin\theta \, \sin\phi \\ \cos\phi \end{bmatrix}.$$

This produces the source and target directions $\boldsymbol{\omega}_s$ and $\boldsymbol{\omega}_t$, whose difference encodes how illumination rotates around the object.

Step 2: SH projection of lighting direction. To obtain a smooth, rotation-aware embedding, we project each direction onto a real spherical-harmonic (SH) basis up to second order ($l \leq 2$), yielding 9 coefficients per light:

$$\mathbf{s}_{\mathrm{SH}}(\boldsymbol{\omega}) = \left[Y_0^0(\boldsymbol{\omega}), Y_1^{-1}(\boldsymbol{\omega}), \ldots, Y_2^2(\boldsymbol{\omega})\right].$$

Low-order SH provides a continuous angular representation in which small yaw–pitch edits correspond to small changes in coefficient space. This avoids the discontinuities and periodic ambiguities associated with raw angle differences.

Step 3: Constructing the relative lighting embedding. Since relighting depends only on *how* illumination changes, not on its absolute parameters, we use the difference between SH encodings:

$$\Delta\mathbf{s}_{\mathrm{SH}} = \mathbf{s}_{\mathrm{SH}}(\boldsymbol{\omega}_t) - \mathbf{s}_{\mathrm{SH}}(\boldsymbol{\omega}_s).$$

We compute photometric and chromatic differences in the same manner: the relative intensity is simply the log-intensity difference $\Delta \log E = \log E_t - \log E_s$, and the relative color temperature is the normalized scalar difference $\Delta\tau = \tau_t - \tau_s$. These quantities measure how much the lighting becomes brighter/dimmer or warmer/ cooler, without encoding their absolute values. The full relative illumination vector is therefore

$$\Delta\ell = \left[\Delta\mathbf{s}_{\mathrm{SH}}, \ \Delta \log E, \ \Delta\tau\right],$$

which is mapped through a small MLP to produce the lighting token t_{light}. This representation provides an intuitive parameterization for user edits while remaining physically interpretable and numerically stable for diffusion training.

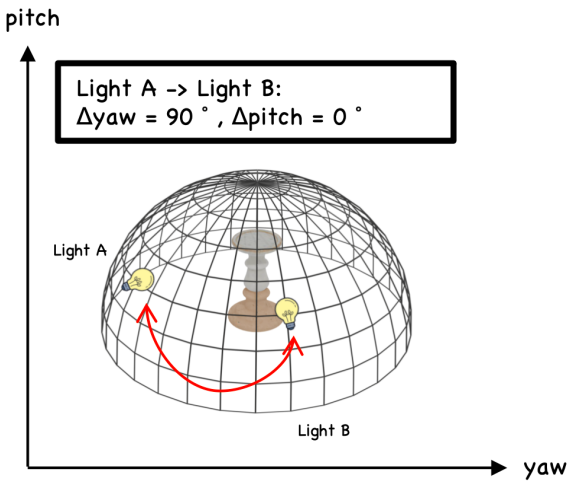

Figure 2. Illustration of our controllable lighting formulation. A lighting edit is first defined in yaw–pitch space (e.g., Light A $\rightarrow$ Light B with $\Delta\text{yaw} = 90^\circ$, $\Delta\text{pitch} = 0^\circ$). These angular parameters are mapped to unit vectors on the hemisphere, then projected onto a low-order spherical-harmonic (SH) basis. The resulting SH difference $\Delta\mathbf{s}_{\text{SH}}$ forms the directional component of our relative illumination encoding $\Delta\ell$.

D. Lighting-aware Mask Design

A key challenge in controllable relighting is that illumination changes are inherently spatially selective: specular highlights shift according to the light direction, shadows migrate across surfaces, and soft shading gradients adjust smoothly, while most of the object's appearance—including albedo and fine geometry cues—remains unchanged. If a diffusion model is trained without spatial guidance, it tends to distribute lighting edits broadly across the image, unintentionally altering material properties or introducing inconsistent shading.

Changes in lighting should not prompt the model to modify intrinsic appearance. However, diffusion models often struggle to distinguish illumination-driven variations from texture changes, especially when trained on diverse lighting setups. A spatial mask offers a practical way to communicate "where the model should change things" versus "where it should stay invariant." This encourages the model to treat material appearance as stable, while letting highlights, cast shadows, and shading gradients adapt fluidly to $\Delta\ell$.

Instead of relying on explicit geometric supervision or perfect shadow segmentation—which would require heavy manual annotation—our mask is derived directly from photometric observations. We compare the luminance fields of the source and target images under logarithmic scaling, which suppresses exposure differences and emphasizes shading variation. A robust, multi-kernel smoothing operation then extracts a soft spatial map that highlights areas where illumination behavior changes noticeably.

This produces a mask M_{gt} that is: smooth enough to guide diffusion without introducing high-frequency artifacts, responsive to both hard shadow boundaries and soft shading shifts, and fully generated without human annotation.

E. Conditioning Architecture and Tokens

Our diffusion backbone follows a U-Net–based architecture with cross-attention layers at each resolution. LightCtrl provides three complementary conditioning signals—appearance, illumination, and proxy cues—which are injected into the model through a unified token-based interface. Each token type is encoded into a fixed-dimensional embedding and consumed by cross-attention during denoising.

Appearance token. The source image $x_s^{\ell_s}$ is first encoded by a frozen vision encoder F_{img} (a CLIP-ViT variant), producing

$$t_{\text{img}} = F_{\text{img}}(x_s^{\ell_s}) \in \mathbb{R}^{1\times d},$$

which captures the coarse geometry and view-dependent texture cues that should remain invariant under relighting. This token serves as the primary identity anchor throughout the diffusion process.

Lighting token. The relative illumination encoding $\Delta\ell$ (Sec. C) is mapped through a lightweight MLP,

$$t_{\text{light}} = \text{MLP}_\ell(\Delta\ell) \in \mathbb{R}^{1\times d},$$

providing a compact representation of directional, energetic, and chromatic lighting changes. This token determines the intended shading and highlight appearance at each denoising timestep.

Proxy (PBR) token. The latent proxy estimated by the PBREncoder, $\hat{\mathcal{B}} = E_\phi(x_s^{\ell_s})$, is spatially pooled and linearly projected:

$$t_{\text{pbr}} = W_{\text{pbr}} \cdot \text{Pool}\big(\hat{\mathcal{B}}\big) \in \mathbb{R}^{1\times d}.$$

This token introduces soft material and geometry priors inferred from sparse PBR supervision, compensating for the under-constrained nature of single-image relighting.

Token fusion in cross-attention. At each U-Net block, the conditioning tokens are concatenated into a unified sequence:

$$T = \big[t_{\text{img}}, t_{\text{light}}, t_{\text{pbr}}\big] \in \mathbb{R}^{3\times d}.$$

Given a latent feature map z_t at diffusion step t, the cross-attention operation is:

$$\mathrm{Attn}(z_t, T) = \mathrm{softmax}\left(\frac{(Qz_t)(KT)^\top}{\sqrt{d}}\right)(VT),$$

where Q, K, V are learned linear projections. This formulation lets the model dynamically select which conditioning stream to attend to at each spatial location. In early layers, t_{img} dominates to preserve object identity, while deeper layers increasingly rely on t_{light} and t_{pbr} to synthesize correct shading, specularity, and global illumination changes.

Mask-weighted fusion. From Sec. D, a spatial mask $M_\theta \in [0,1]^{H\times W}$ identifies illumination-sensitive regions. We therefore modulate cross-attention using:

$$\begin{aligned}\mathrm{Attn}^\star(z_t, T) = (1 - M_\theta) \odot \mathrm{Attn}_{\mathrm{base}}(z_t, t_{\mathrm{img}}) \\ + M_\theta \odot \mathrm{Attn}_{\mathrm{full}}(z_t, T).\end{aligned}$$

so that stable regions rely primarily on appearance cues while illumination- variant areas receive full conditioning. This mechanism prevents unnecessary texture distortion and preserves material fidelity under large lighting edits.

Overall, the combination of appearance, lighting, and proxy tokens enables LightCtrl to achieve fine-grained, disentangled relighting while maintaining strong geometric and photometric consistency.

F. Training Details

All experiments are conducted on 4×NVIDIA H800 GPUs using mixed-precision (fp16) training. We optimize LightCtrl with the AdamW optimizer, using an initial learning rate of $\mathbf{1 \times 10^{-4}}$, β_1=0.9, β_2=0.999, and a weight decay of 1×10^{-2}. The batch size is set to 16 per GPU (64 total), and the model is trained for 400K diffusion steps using a constant-with-warmup schedule with 5K warm-up steps. We adopt a DDIM noise scheduler with 1000 timesteps and follow the standard v-prediction objective. Unless otherwise stated, all models are selected using the best validation performance on the held-out ScaLight split.

We also evaluated LightCtrl on one NVIDIA H800(CUDA 12.2) using **bf16 precision** optimization. Results demonstrate high efficiency: the model (∼1.2B parameters) generates a 512 × 512 image in just ∼**0.84s** (50 steps) with a peak memory usage of **2.53GB**.

G. Implementation Details of the DPO Mechanism

Our pipeline initializes with joint training using large-scale relighting pairs without ground-truth (GT) intrinsic supervision. While this implicit learning establishes strong generative priors, it can occasionally entangle material and lighting representations. To robustify the latent proxy encoder, we introduce a post-training stage using sparse PBR supervision, where only ∼3% of the training data contains explicit annotations. Under such extreme label sparsity, we find that standard Supervised Fine-Tuning (SFT) suffers from severe generalization issues. Specifically, SFT merely demonstrates the "correct" state and often leads the model to overfit, causing it to memorize lighting artifacts such as baking cast shadows into the predicted albedo.

To overcome this, we adopt Direct Preference Optimization (DPO) over SFT. DPO leverages a contrastive signal, teaching the model not just what constitutes valid physical consistency, but crucially, what destroys it. Implementation-wise, we construct preference pairs dynamically during training. We define the preferred positive sample y_{pos} using the GT PBR maps. For the rejected negative sample y_{neg}, we use the model's own current prediction (i.e., $y_{neg} = E_\phi(x_s^{\ell_s})$). Because y_{neg} is generated by the network itself, it inherently contains the specific artifacts and entangled shading that the model is prone to making. By constructing pairs in this manner, the training process explicitly penalizes the exact failure modes the model naturally produces.

H. Impact of PBR Supervision Scale

To quantify how the amount of PBR-supervised data influences the quality of our latent proxy and overall relighting performance, we conduct a controlled study using different sizes of PBR annotations while keeping all other training settings fixed. Specifically, we train LightCtrl with {1K, 3K, 6K, 9K} PBR-supervised samples randomly drawn from ScaLight, and evaluate on the same held-out object-level test set. Another 3K samples is collected in the Objaverse datasets.

Since the PBR encoder is the primary source of material and geometry priors, increasing the number of PBR-supervised examples naturally improves the fidelity of shading transfer and relighting accuracy. However, as shown in Table 2, the gains diminish beyond roughly 9K supervised samples, indicating clear marginal returns. This saturation suggests that dense intrinsic annotations are unnecessary: even relatively sparse supervision is sufficient to provide stable material cues for the diffusion model and yield high-quality, controllable relighting.

I. Additional Qualitative Results

For completeness, we provide additional qualitative examples that complement the results shown in the main paper. Figures 3, 4, 5, 6, 7, 8, 9 and 10 illustrate more relighting outcomes under controlled edits of (i) light position (yaw–pitch), (ii) illumination intensity, and (iii) color temperature. These samples demonstrate that LightCtrl produces smooth, photometrically consistent transitions across

Table 2. Effect of PBR supervision scale on relighting quality. Increasing PBR annotations steadily improves performance, yet our experiments show that even sparse supervision provides meaningful benefits for relighting tasks.

#PBR Samples	RMSE ↓	SSIM ↑	PSNR ↑
1K	0.468	0.487	12.8
3K	0.421	0.534	14.7
6K	0.331	0.612	19.3
9K	0.194	0.873	23.5
12K	0.105	0.921	28.6

continuous lighting variations. Finally, additional figures present additional scene-level results on both synthetic environments and real-world images from MIIW.

J. Conclusion

In this supplementary document, we provided additional technical details of LightCtrl, including the formulation of our relative illumination encoding, the construction and training of the lighting-aware mask, and the integration of proxy, appearance, and lighting tokens within the conditioning architecture. We further described implementation specifics such as training schedules and hardware, and presented additional qualitative examples covering light position, intensity, color temperature, and scene-level relighting. Together, these materials complement the main paper by offering deeper insight into the design choices behind LightCtrl and by demonstrating its robustness under a wide range of controlled illumination conditions.

References

[1] Zexin He, Tengfei Wang, Xin Huang, Xingang Pan, and Ziwei Liu. Neural lightrig: Unlocking accurate object normal and material estimation with multi-light diffusion. In *CVPR*, 2025.

[2] Xiaoyan Xing, Konrad Groh, Sezer Karaoglu, Theo Gevers, and Anand Bhattad. Luminet: Latent intrinsics meets diffusion models for indoor scene relighting. In *CVPR*, 2025.

[3] Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. Rgb-x: Image decomposition and synthesis using material- and lighting-aware diffusion models. In *ACM SIGGRAPH*, 2024.

[4] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Scaling in-the-wild training for diffusion-based illumination harmonization and editing by imposing consistent light transport. In *ICLR*, 2025.

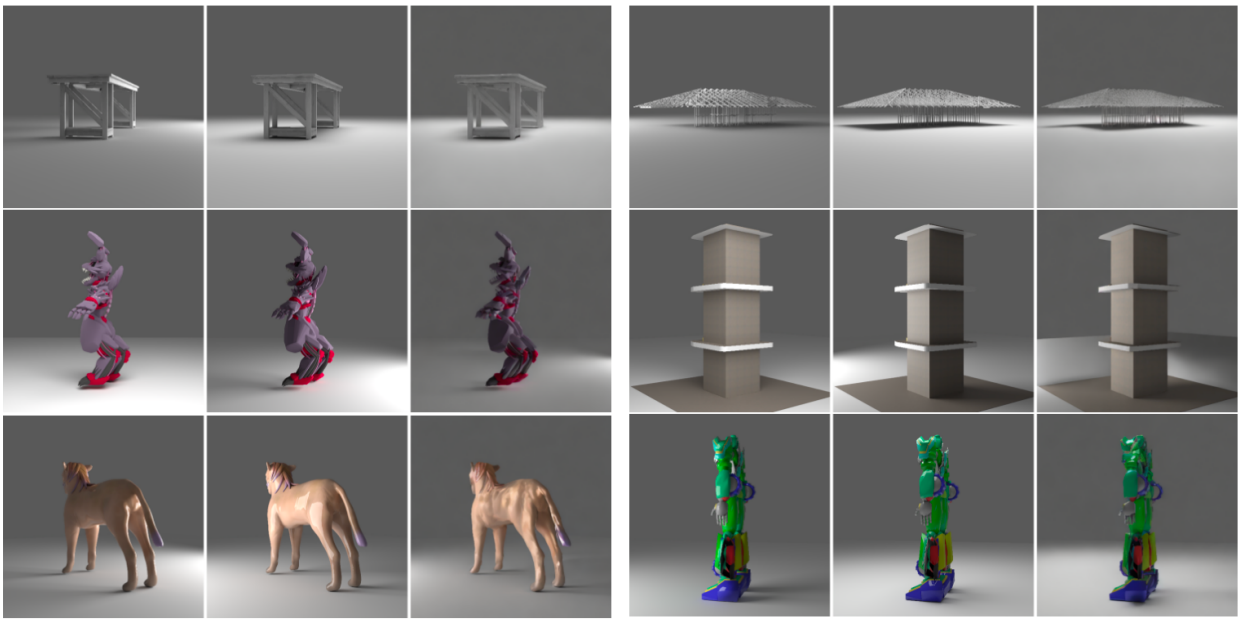

Figure 3. Qualitative relighting results on **ScaLight** under varying **light positions**. Left: reference image; middle: ground-truth relit image; right: **LightCtrl** prediction.

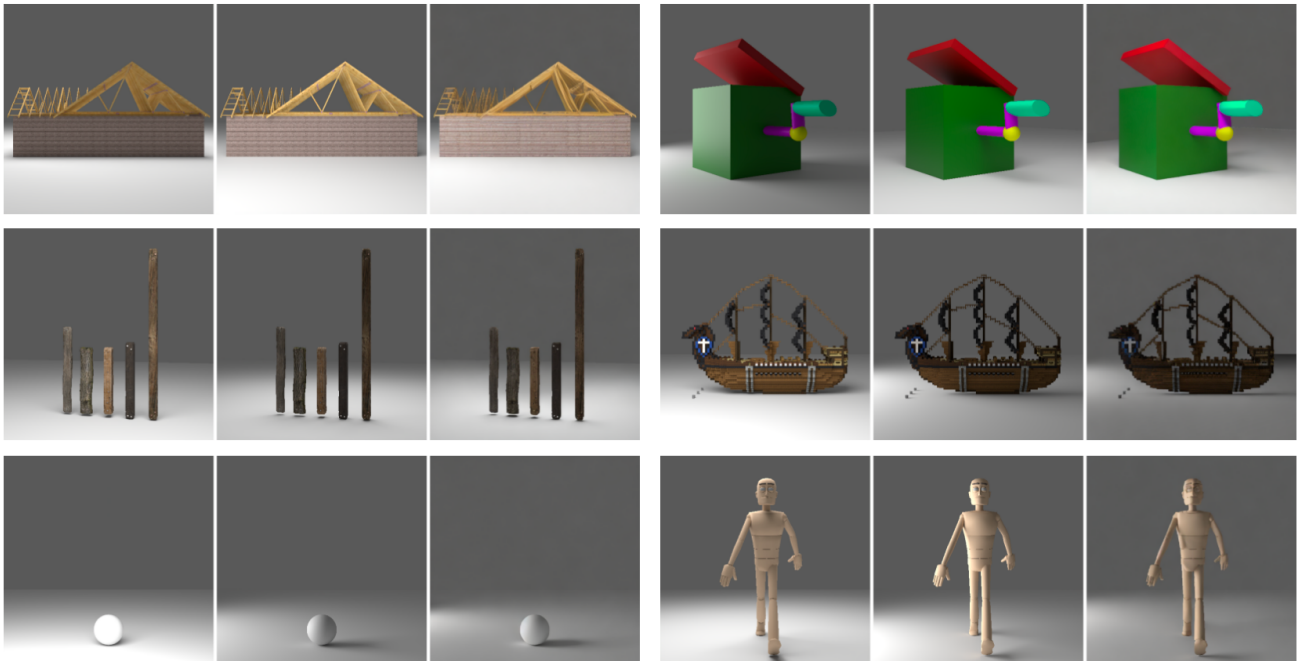

Figure 4. Qualitative relighting results on **ScaLight** under varying **light positions**.

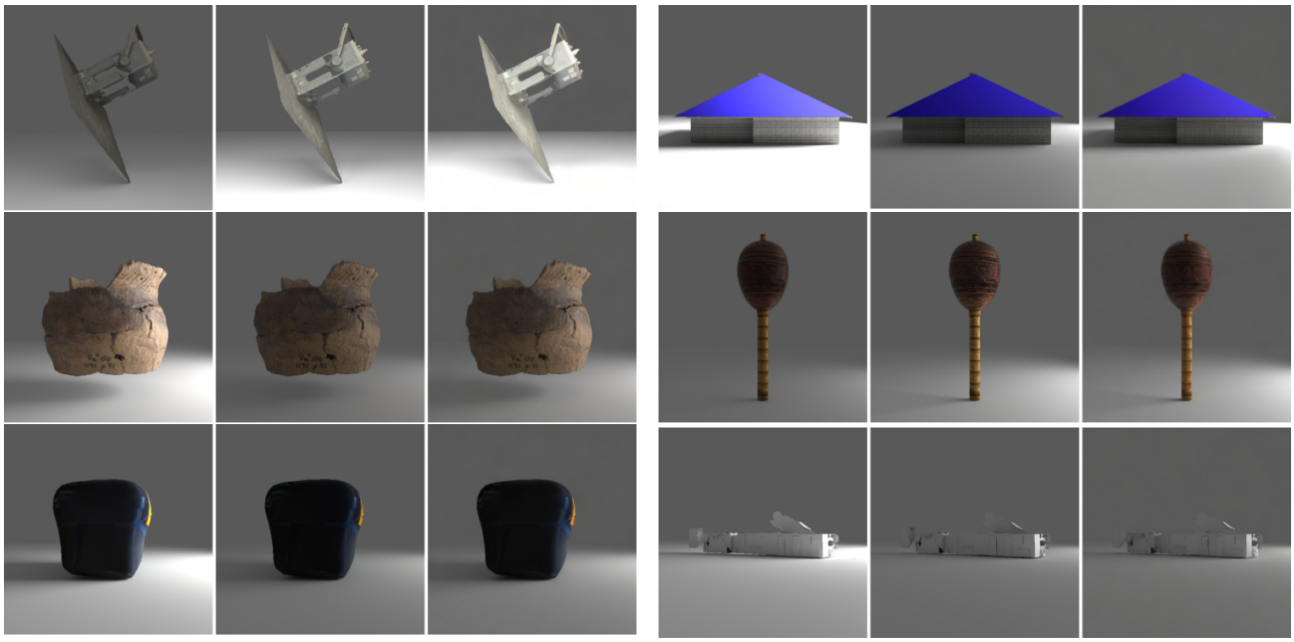

Figure 5. Qualitative relighting results on **ScaLight** under varying **light intensity**. Left: reference image; middle: ground-truth relit image; right: **LightCtrl** prediction.

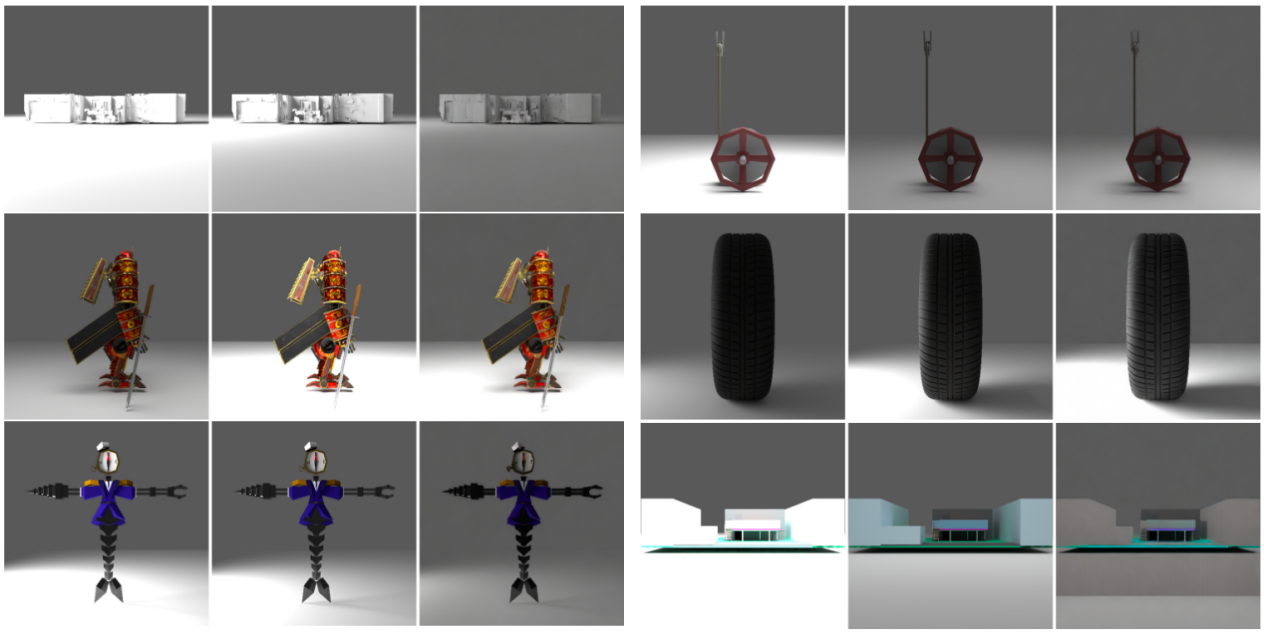

Figure 6. Qualitative relighting results on **ScaLight** under varying **light intensity**.

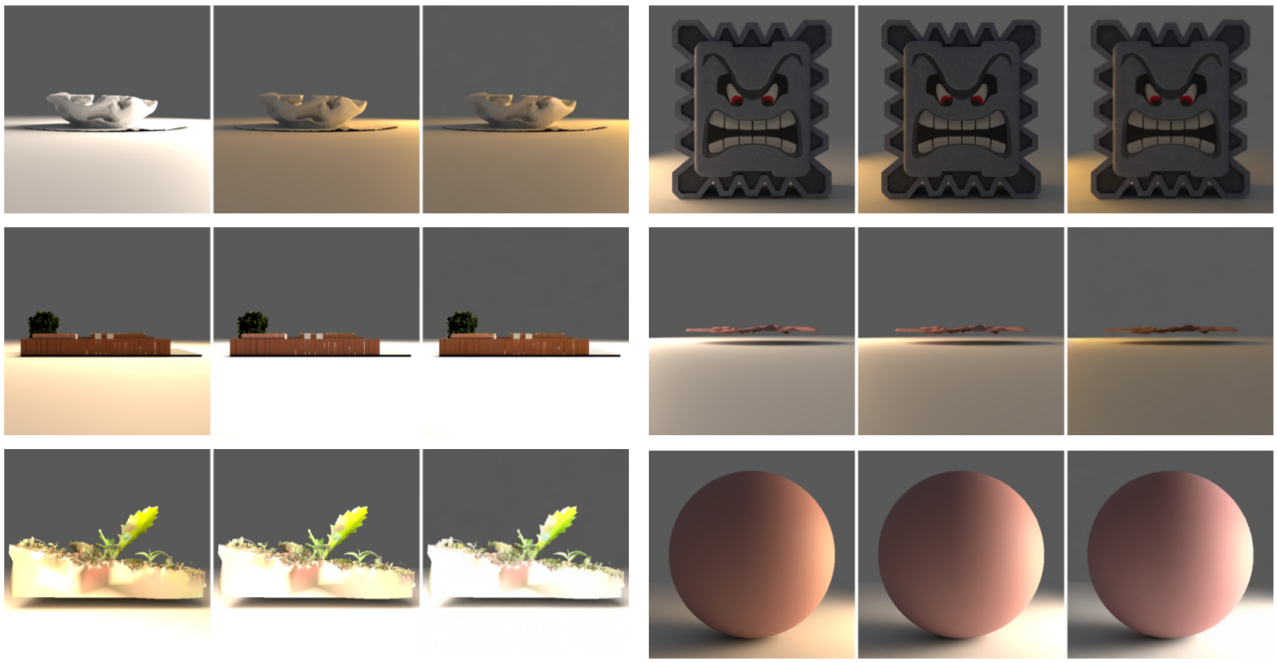

Figure 7. Qualitative relighting results on **ScaLight** under varying **light temperature**. Left: reference image; middle: ground-truth relit image; right: **LightCtrl** prediction.

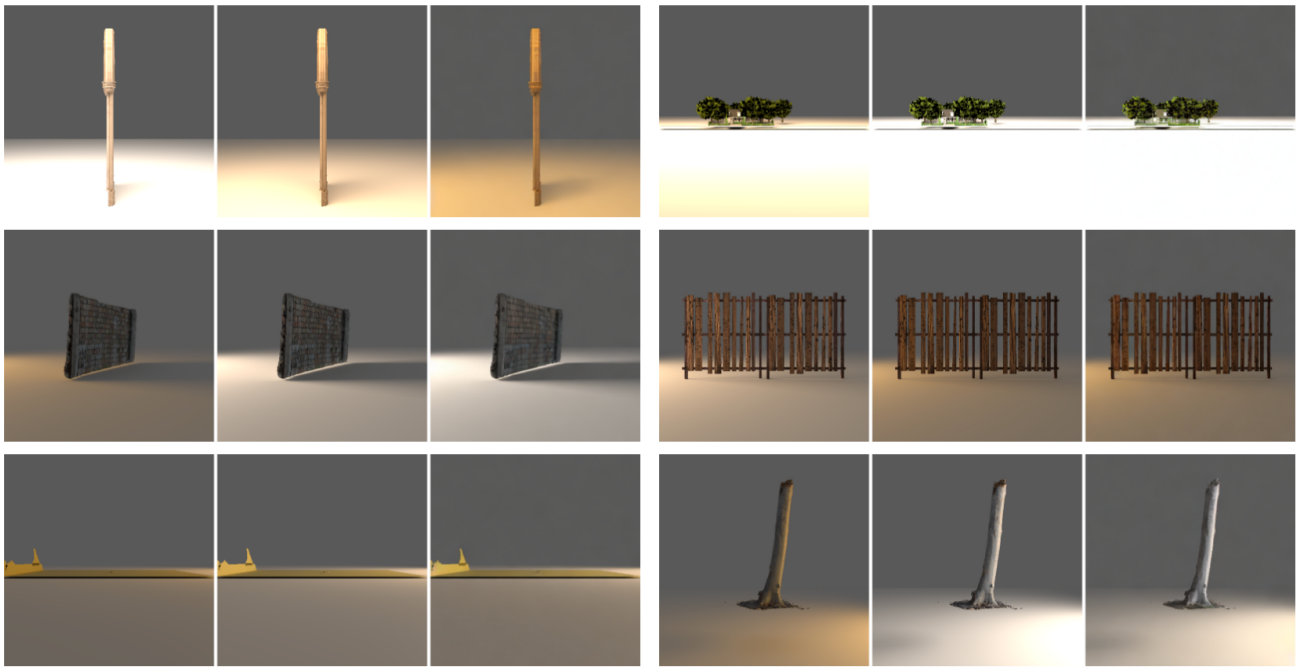

Figure 8. Qualitative relighting results on **ScaLight** under varying **light temperature**

Figure 9. Qualitative relighting results on **MIIW**. Left: reference image; middle: ground-truth relit image; right: **LightCtrl** prediction.

Figure 10. Qualitative relighting results on **MIIW**. Left: reference image; middle: ground-truth relit image; right: **LightCtrl** prediction.