SyncLight: Single-Edit Multi-View Relighting

David Serrano-Lozano[1,2] **Anand Bhattad**[3,†] **Luis Herranz**[4]
Jean-François Lalonde[5,†] **Javier Vazquez-Corral**[1,2,†]

[1]Computer Vision Center [2]Universitat Autònoma de Barcelona [3]Johns Hopkins University
[4]Universidad Politécnica de Madrid [5]Université Laval [†]Equal advising

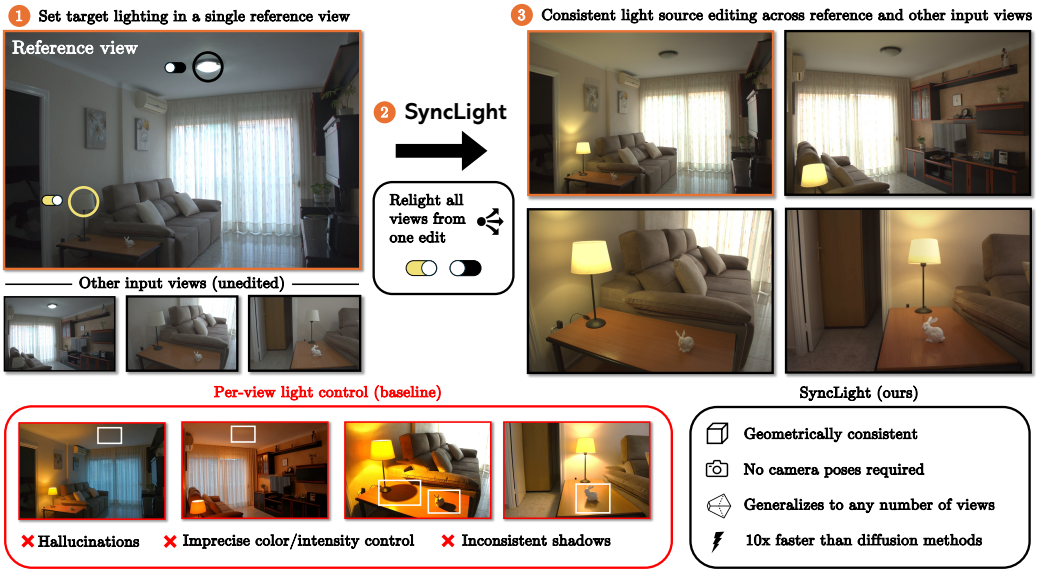


Figure 1: Multi-view light editing with SyncLight. Given an uncalibrated multi-view capture of a static scene (1), the user picks a *reference view*, clicks one or more visible light sources (circle markers), and sets their intensity and chromaticity; the other views never need to be inspected. SyncLight (2) relights all input views, reference included (3), in a *single* feedforward pass via latent bridge matching. Per-view light control baselines (bottom-left) hallucinate, lack precise control, and are inconsistent across views. SyncLight is geometrically consistent, needs no camera poses, generalizes to any number of views, and is 10× faster than current models.

Abstract

We present **SyncLight**, a method to enable consistent, parametric control over light sources across multiple uncalibrated views of a static scene conditioned on a single view. While single-view relighting has advanced significantly, existing generative approaches struggle to maintain the rigorous lighting consistency essential for multi-camera broadcasts, stereoscopic cinema, and virtual production. SyncLight addresses this by enabling precise control over light intensity and color across a multi-view capture of a scene, conditioned on a single reference edit. Our method leverages a multi-view diffusion transformer trained using a latent bridge matching formulation, achieving high-fidelity relighting of the entire image set in a single inference step. To facilitate training, we introduce a large-scale hybrid dataset comprising diverse synthetic environments—curated from existing sources and newly designed scenes—alongside high-fidelity, real-world multi-view captures under calibrated illumination. Though trained only on image pairs, SyncLight generalizes zero-shot to an arbitrary number of viewpoints, effectively propagating lighting changes across all views, without requiring camera pose information. SyncLight enables practical relighting workflows for multi-view capture systems. https://color.cvc.uab.cat/synclight/

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Controlling illumination after capture in multi-view settings remains an open problem, even as neural rendering [51, 32] has made high-fidelity 3D geometry capture increasingly accessible. This is especially difficult for indoor scenes, where multiple light sources, interreflections, and occlusions create a rich illumination landscape—and where professional workflows like stereoscopic cinema and virtual production demand rigorous consistency, not just plausible appearance.

Relighting is fundamentally an ill-posed problem: from captured images alone, one cannot uniquely decompose appearance into geometry, material, and lighting. State-of-the-art generative methods like LightLab [49] and ScribbleLight [10] achieve photorealistic single-image editing, but process each view independently and hallucinate lighting effects that are locally plausible yet globally contradictory: shadows change direction between cameras, specular highlights remain stationary as viewpoints shift. Conversely, inverse rendering and NeRF-based approaches [73, 27] enforce physical consistency through explicit geometry estimation, but require known camera poses, expensive per-scene optimization, and struggle in regions where geometric estimates fail.

To resolve this dichotomy, we introduce **SyncLight**, a *pose-free* generative framework that brings the flexibility of 2D image editing to multi-view indoor scene relighting. Unlike prior multi-view approaches that depend on calibrated cameras or explicit 3D reconstruction, SyncLight operates directly on uncalibrated image sets: a multi-view diffusion transformer learns to discover cross-view correspondences implicitly through attention, sidestepping both the ill-posed decomposition and the brittle reliance on geometric supervision. A user adjusts light intensity and color on a single reference image, and SyncLight propagates these changes across all synchronized viewpoints in a single forward pass, preserving the complex interplay of shadows, highlights, and indirect illumination.

Realizing this system requires addressing two challenges. First, no existing dataset provides multi-view captures with calibrated lighting ground truth; we introduce a large hybrid dataset combining procedurally generated synthetic environments with high-fidelity real-world captures. Second, prior video-based approaches require per-frame lighting specification, making editing tedious; our multi-view transformer instead propagates a single reference edit jointly to all views, again without any pose information. A key emergent property is *zero-shot scalability*: although trained exclusively on image pairs ($N = 2$), our attention mechanism generalizes to dense camera arrays ($N > 2$) at inference without retraining. Fig. 1 illustrates this workflow, where users click on the reference image to control intensity and chromaticity, and the edits propagate consistently across all views.

In summary, we make the following contributions:

- **Pose-free multi-view relighting:** The first generative method to parametrically relight synchronized, uncalibrated views with strict spatial and photometric consistency in a single forward pass—without any camera poses or explicit 3D reconstruction.
- **Zero-shot generalization:** A relighting-aware Multi-View Transformer that learns consistency from image pairs while scaling to arbitrary view counts at inference.
- **Efficient one-step inference:** Bridge matching delivers high-fidelity results in a single pass, providing a production-ready alternative to slow optimization baselines.
- **SyncLight dataset:** A comprehensive dataset of multi-view indoor scenes under varying illumination, bridging synthetic and real-world domains.

2 Related Work

The paradigm of relighting has recently shifted from physically-based inverse rendering [38, 37, 67, 40, 74, 28, 29, 19, 18, 39] to data-driven generative modeling. Rather than explicitly solving the ill-posed decomposition of appearance into geometry, materials, and lighting, modern approaches learn to hallucinate plausible lighting transformations directly in image space [7, 70, 62, 49], leveraging the fact that such decompositions can emerge implicitly within generative models [6, 14].

Domain-specific methods. Early generative approaches focused on constrained settings. Portrait relighting methods leverage light stage captures [23, 50] to model facial appearance under controlled illumination, but are limited to frontal faces. Object-centric approaches [65, 13, 26, 5, 71, 17, 3] condition diffusion models on environment or intrinsic maps, spherical harmonics, or shadow masks, enabling material-aware relighting for isolated objects. However, these methods do not generalize well to complex indoor scenes with multiple interacting light sources and global illumination.

Single-view diffusion for general indoor scenes. Recent diffusion-based methods achieve remarkably photorealism by framing relighting as conditional synthesis. IC-Light [68] leverages large 2D datasets to hallucinate lighting effects based on text or environment maps, while LumiNet [62] and StyLitGAN [7] explore unsupervised latent control [70] but suffer from intrinsic ambiguities. Methods such as IntrinsicEdit [48], IID [35], LightIt [34], PractiLight [15], and RGB↔X [66] use intrinsic representations for relighting, though they require explicit maps or lack spatial control.

ScribbleLight [10] and LightLab [49] are state-of-the-art in single-view relighting. Both process views independently and cannot maintain cross-camera consistency. SyncLight instead uses cross-view attention to propagate a single reference edit across all views in a single forward pass, requiring no per-view masks, depth, or camera poses. Trained only on stereo pairs, it generalizes zero-shot to larger camera arrays at inference. We further replace iterative diffusion (50+ steps) with Latent Bridge Matching for one-step inference (10–50× speedup).

3D-consistent approaches. NeRF and 3DGS-based methods [73, 27, 3, 54, 45] and inverse rendering approaches enforce consistency through explicit 3D decomposition into geometry, materials, and lighting. While these ensure geometric consistency, they require per-scene optimization and lack flexibility for rapid editing. More recently, [8, 43] combine monocular geometry estimation with physically-based rendering to enable CG-like control over light placement in scenes. While this achieves physical accuracy, it requires explicit mesh reconstruction and does not address multi-view consistency constraints of synchronized camera arrays. SyncLight instead learns consistency priors directly from multi-view data using a generative prior without per-scene 3D reconstruction.

Multi-view generation and consistency. Recent work has shown that explicit cross-view attention is effective for enforcing geometric consistency. MVDream [57] introduced multi-view diffusion transformers for text-to-3D generation, demonstrating that shared attention across views yields consistent outputs. We build on this insight for relighting, showing that multi-view transformers trained on stereo pairs generalize zero-shot to larger camera arrays and wide-baseline stereo images never seen during training. Temporal consistency in video—conceptually analogous to spatial cross-view consistency—has been studied in recent video relighting work. RelightVid [16] and UniRelight [22] mitigate flickering via temporal modeling and joint denoising. However, these methods operate on sequential frames and do not enforce rigid geometric consistency across viewpoints.

GR3EN [63] and the concurrent LuxRemix [42] also tackle multi-view indoor relighting, but differ considerably from our approach. LuxRemix performs per-view generative OLAT *decomposition* and harmonizes the decomposed lights into a relightable 3DGS representation. GR3EN instead requires *posed* multi-view inputs, reconstructs a radiance field, and distills a video-to-video relighting diffusion model back into the 3D model. Both therefore rely on an explicit per-scene 3D reconstruction, and GR3EN additionally requires known camera poses. SyncLight, in contrast, is pose-free and purely 2D: cross-view consistency emerges from our multi-view transformer, and the entire image set is relit in a single feedforward pass without any per-scene optimization.

Flow matching and efficient sampling. Our use of Latent Bridge Matching [9] builds on recent advances in flow-based generative models. Flow Matching [44] introduced simulation-free training of continuous normalizing flows, enabling faster sampling than standard diffusion. Bridge Matching and Schrödinger Bridge methods [2, 58] further refine these approaches by optimizing transport between arbitrary distributions by adding stochasticity in the trajectory. We adopt this formulation to enable one-step inference essential for interactive multi-camera relighting workflows.

3 Method

Our goal is to enable consistent, parametric light-source control across multiple synchronized views by conditioning on parametric lighting edits applied to a single reference view. Unlike prior multi-view diffusion frameworks such as MVDream [57] or Wonder3D [47], which target image-to-3D generation and require explicit camera pose information, SyncLight is a pose-free, generative framework for editing light sources in synchronized views of an existing scene. Our model implicitly discovers cross-view correspondences through attention, without any geometric supervision, by specifying only the desired light conditions in one reference view. We build upon latent diffusion models [56] and extend them to the multi-view setting through two key contributions: (1) a multi-view transformer with cross-view attention that enforces spatial coherence and consistent propagation of lighting edits, and (2) LBM formulation [9] that enables one-step inference, avoiding the iterative sampling required by standard diffusion models. Fig. 2 provides an overview of our approach.

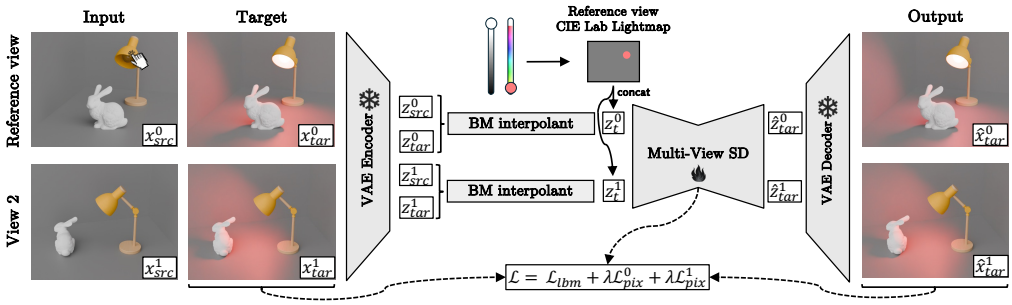


Figure 2: SyncLight formulates multi-view relighting as a conditional flow matching problem in latent space. (Left) Input scenes under source ($\mathbf{x}_{\text{src}}$) and target lighting ground truth ($\mathbf{x}_{\text{tar}}$) are encoded into latents ($\mathbf{z}_{\text{src}}, \mathbf{z}_{\text{tar}}$) via a VAE encoder. This is done for both the "Reference" (0) and "Other" (1) views. (Middle) During training, we sample a timestep t and construct a Bridge Matching (BM) interpolant $\mathbf{z}_t$. Our backbone, "Multi-View SD", is conditioned on a user-specified "Lightmap" (encoding color as intensity and chromaticity) derived from the "Reference view". To enforce consistency, the backbone processes both views simultaneously. (Right) The model predicts the target latents, which are decoded into relit images ($\hat{\mathbf{x}}_T$). The network is optimized using a hybrid objective $\mathcal{L}$ combining latent flow matching loss ($\mathcal{L}_{\text{lbm}}$) with pixel-level reconstruction losses ($\mathcal{L}_{\text{pix}}$) for each view to ensure high-fidelity, consistent relighting.

Latent bridge matching. Traditional latent diffusion models learn to denoise from pure Gaussian noise, which is inefficient for image-to-image translation tasks, where the source image already contains most of the desired scene content. Flow matching [44] provides a framework for learning continuous transport between arbitrary distributions. We adopt LBM [9], which constructs flows between a source latent $\mathbf{z}_{src}$ (encoding an image under source lighting) and a target latent $\mathbf{z}_{tar}$ (encoding the same scene under target lighting).

We encode an image under source and target lighting $\{\mathbf{x}_{\text{src}}, \mathbf{x}_{\text{tar}}\}$ into latent representations using a pretrained VAE encoder, $\mathbf{z} = \mathcal{E}(\mathbf{x})$, and define a stochastic interpolation path between them:

$$\mathbf{z}_t = (1-t)\mathbf{z}_{\text{src}} + t\mathbf{z}_{\text{tar}} + \sigma\sqrt{t(1-t)}\boldsymbol{\epsilon}\,, \tag{1}$$

where $t \in [0,1]$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, and σ controls the noise magnitude. Following prior work [46, 9, 59], we use $\sigma = 0.005$, resulting in stochastic trajectories.

The model $\mathbf{v}_\theta$ is trained to predict the velocity field that transports $\mathbf{z}_{\text{src}}$ to $\mathbf{z}_{\text{tar}}$. The target velocity is:

$$\mathbf{v} = \mathbf{z}_{\text{tar}} - \mathbf{z}_{\text{src}}. \tag{2}$$

The training objective minimizes the mean squared error between predicted and target velocities:

$$\mathcal{L}_{\text{lbm}} = \mathbb{E}_{t,\mathbf{z}_{\text{src}},\mathbf{z}_{\text{tar}}}\left[\left|\left|\frac{\mathbf{v}_\theta(\mathbf{z}_t, t, c)}{(1-t)} - \mathbf{v}\right|\right|_2^2\right], \tag{3}$$

where c is the conditioning information. We limit t to only 4 equally-spaced timesteps during training.

At inference time, given a source latent $\mathbf{z}_{\text{src}}$ and a conditioning c, the target latent is directly estimated in a *single* step. The decoded output $\hat{\mathbf{x}}_{tar} = \mathcal{D}(\hat{\mathbf{z}}_{tar})$ provides the relit image. Compared to iterative diffusion sampling (50+ steps), this one-step inference is significantly faster while maintaining high visual quality, making it practical for interactive multi-view applications. Observe that at $t = 0$, eq. (3) collapses to an L_2 loss between $\mathbf{v}_\theta$ and the residual $\mathbf{v}$, which might suggest that the LBM formulation reduces to standard regression under one-step inference. However, this is not the case: training across equally-spaced timesteps t forces the network to learn a well-conditioned velocity field along the entire stochastic interpolant, rather than a brittle point-to-point mapping; in line with the observations of [44], this multi-timestep supervision yields more generalizable transport even when inference is truncated to a single step. See Supplementary Material for further discussion.

Multi-view latent bridge matching. SyncLight takes as input two synchronized, uncalibrated views under source lighting $\{\mathbf{x}^0_{\text{src}}, \mathbf{x}^1_{\text{src}}\}$ and a lighting condition c^0 defined on the reference view $\mathbf{x}^0_{\text{src}}$, and predicts both views under target lighting $\{\hat{\mathbf{x}}^0_{\text{tar}}, \hat{\mathbf{x}}^1_{\text{tar}}\}$. Standard diffusion and flow-based architectures are designed for single-image inputs and do not model cross-view interactions. We therefore use Stable Diffusion XL [53] as our velocity field model and adapt it to explicitly reason across multiple views while using its pretrained weights as the model initialization.

Inspired by MVDream [57], we replace each transformer block in SDXL with a multi-view transformer block. The key idea is to enable information exchange across N different views through modified self-attention. Given a batch of N view latents, we first process each view's features independently through the initial convolution layers. Before each transformer block, features from all views are concatenated along the token dimension rather than the batch dimension. This allows the self-attention mechanism to attend across views, capturing cross-view geometric and photometric consistency. After attention, features are reshaped back to the original per-view layout. This rearrangement is essential. In the SDXL UNet, views stacked along the batch dimension never interact in the attention layers, so naively feeding several views to a single-view model does not propagate the edit at all. Although we train exclusively on pairs ($N = 2$), the architecture is agnostic to the number of views due to the token-level attention mechanism. At inference, we observe generalization to arbitrary view counts while maintaining consistency (see sec. 5). Formally, the input to the model $\mathbf{v}_\theta$ is the batch-wise concatenation of all interpolated latent views: $\mathbf{z}_t = \text{concat}[\mathbf{z}_t^0, \cdots, \mathbf{z}_t^{N-1}]$. Despite this architectural inspiration, we note that our setting differs fundamentally: MVDream and related works (e.g., Wonder3D [47]) are image-to-3D *generative* models that require camera pose conditioning, whereas SyncLight is a pose-free parametric light source *editing* model operating on synchronous views.

Lighting control. We represent target lighting conditions via a 4-channel lightmap $L \in [-1, 1]^{H \times W \times 4}$ defined on the reference view $\mathbf{x}_{\text{src}}^0$. The only constraint on the reference view is that the light sources to be edited must be visible. Users specify locations of light sources to modify (via circular markers), and the lightmap encodes: (1) activation state (1 = on, -1 = off), and (2–4) target color in Lab space. All values default to 0, unless the user specifies the activation and color. We use Lab rather than RGB to enable independent control of brightness (L) and chromaticity (ab). The lightmap is resized to latent resolution and concatenated to each interpolated latent $\mathbf{z}_t^i$, yielding 8-channel inputs (4 latent + 4 lightmap). Since the pretrained SDXL backbone expects 4-channel latent inputs, we replace its first convolutional layer with an 8-channel variant whose four additional input channels (corresponding to the lightmap) are zero-initialized. Since all latents must have the same dimensions, we apply the same lightmap to all views—this works well in practice, likely because providing color and intensity hints to other views is helpful, even despite the spatial misalignment. The model learns that the lightmap lives in reference-view coordinates and associates the marked fixture across views; performance does not degrade with the inter-view angle (fig. 6).

The interaction is thus one click, plus a color and intensity, on a single reference image per edited light. The user never inspects the other views. SyncLight resolves visibility and occlusion itself, even where the edited source is invisible (fig. 7). In contrast, LightLab [49] and GR3EN [63] need a segmentation mask in every frame (GR3EN also needs camera poses), and LuxRemix [42] needs Plücker embeddings and a harmonization stage (tab. 10).

Training objective. In addition to the velocity-matching objective ($\mathcal{L}_{\text{lbm}}$ in eq. (3)), we follow prior work [9, 64] and impose reconstruction losses on the decoded images:

$$\mathcal{L} = \mathcal{L}_{\text{lbm}} + \lambda \mathcal{L}_{\text{pix}}^0 + \lambda \mathcal{L}_{\text{pix}}^1 , \tag{4}$$

where $\mathcal{L}_{\text{pix}}^i$ denotes the LPIPS loss [69] for view i. This perceptual LPIPS penalizes the blur-averaging failure mode, encouraging the network to produce confident, crisp predictions of shadow boundaries.

4 Dataset

Training a generative model for consistent multi-view relighting requires data satisfying three strict criteria: (1) diverse, high-fidelity indoor geometry; (2) perfectly synchronized multi-view camera poses; and (3) isolated, controllable light sources for supervision. Existing datasets fail to meet these requirements. To address this gap, we introduce the **SyncLight Dataset**, comprising nearly one million multi-view image pairs under diverse lighting conditions, designed to enable learning of physically consistent relighting across viewpoints.

Following prior work [25, 1, 49, 21], we employ a One Light At a Time (OLAT) capture scheme: for each scene, we render or photograph each controllable light source independently, together with an ambient capture containing only non-controllable illumination. All images are captured in linear color space (HDR for synthetic data, camera RAW for real captures) to ensure the photometric linearity required for accurate light composition. We then generate arbitrary lighting conditions by linearly

Table 1: Quantitative multi-view image relighting results on our SyncLight test set. Each section of the table indicates which view is used to compute metrics: the "Reference (Ref.) view" (where edits are specified), "Second (Sec.) view" (another view in the set), and "Additional (Add.) views" (all but the reference view). Best results, achieved by SyncLight in all cases, are highlighted. See text for details on baselines. As SyncLight is always run on more than one image; no inference time is reported for the "Ref. view." VGGTm (%) measures cross-view consistency against the ground truth (see text); it is undefined for the reference view alone.

		Infinigen				BlenderKit				Real captures					
	Method	PSNR	SSIM	ΔE_{00}	LPIPS	PSNR	SSIM	ΔE_{00}	LPIPS	PSNR	SSIM	ΔE_{00}	LPIPS	VGGTm	Time (s)
Ref. view	ScribbleLight	10.89	.518	30.67	.382	11.02	.528	29.26	.372	11.78	.483	28.36	.375	-	58.2 ± 1.14
	Flux.2-dev	14.23	.672	20.31	.338	16.71	.704	17.22	.291	18.32	.757	14.28	.264	-	55.6 ± 1.02
	LightLab*	29.86	.941	3.08	.131	26.38	.907	4.92	.147	28.42	.880	4.61	.199	-	1.17 ± 0.02
	GR3EN	25.92	.893	4.38	.158	21.74	.803	8.93	.173	22.31	.782	9.31	.238	-	72.3 ± 3.51
	SyncLight	31.32	.950	2.47	.119	27.16	.915	4.02	.134	30.34	.895	3.73	.196	-	-
Sec. view	Scribble+LNet	13.41	.498	19.67	.284	12.72	.507	23.91	.306	14.48	.518	23.77	.318	50.1	69.6 ± 1.82
	Flux.2-dev	12.75	.617	25.82	.386	14.92	.604	20.40	.469	16.35	.685	19.13	.305	52.7	111 ± 2.08
	LightLab*+LNet	17.36	.817	7.89	.184	15.92	.863	9.82	.196	20.46	.815	8.46	.279	68.3	13.1 ± 0.68
	GR3EN	25.88	.891	4.47	.163	21.43	.800	9.08	.178	22.03	.772	9.52	.244	82.7	76.4 ± 3.67
	SyncLight	31.45	.949	2.45	.119	28.19	.925	3.72	.126	30.23	.897	3.56	.197	95.8	1.58 ± 0.02
Add. views	Scribble+LNet	13.36	.494	19.65	.292	12.88	.509	23.47	.299	14.29	.525	23.92	.320	49.8	89.7 ± 1.93
	Flux.2-dev	12.55	.616	26.03	.388	14.90	.606	20.03	.472	16.21	.678	20.42	.328	54.1	273 ± 3.87
	LightLab*+LNet	17.22	.808	8.35	.195	16.07	.871	10.43	.185	20.40	.809	8.93	.287	66.8	38.5 ± 0.83
	GR3EN	25.94	.894	4.32	.160	21.55	.804	8.94	.174	21.87	.769	9.72	.246	82.1	91.9 ± 3.91
	SyncLight	31.42	.941	2.57	.122	28.01	.929	3.78	.125	30.44	.905	3.57	.194	94.9	2.38 ± 0.03

combining these captures with per-light color and intensity coefficients sampled in Lab space. We impose the constraint that the lights being modified must be visible in at least one view, set as the *reference* view, while the second view may or may not have direct visibility of the modified light sources. For our control, we generate a lightmap for the reference view: a mask with colored circles placed at the spatial location of each modified light source, encoding the target color and intensity.

Our dataset comprises three sources with complementary objectives: (i) 365 *Infinigen* [55] scenes to help the model leverage pretrained scene geometry priors; (ii) 40 *BlenderKit*[†] scenes to provide higher realism, furniture, and light sources; and (iii) *Real* captures from 40 scenes to bridge the domain gap between synthetic and real-world data. In total, the SyncLight Dataset contains nearly one million multi-view training pairs: 920,000 from Infinigen, 47,000 from BlenderKit, and 18,000 from real captures. For testing, we hold out a diverse subset of scenes not seen during training. Detailed capture protocols, light composition equations, and exemplar images for each split are provided in the Supplementary.

5 Experiments

Results and comparisons. For single-view relighting of the reference view, we compare against ScribbleLight [10] using Marigold [30, 31] to obtain the albedo and depth, and our own implementation of LightLab [49] (**LightLab***). Given that the original model and data are not publicly available, we closely follow the reported design, including a fine-tuned SDXL backbone conditioned on parametric light masks and the original single-view training objective. To ensure a fair comparison, we train our reproduction on the same dataset used for SyncLight, excluding the multi-view component, allowing us to isolate the effect of multi-view modeling. LightLab* serves as our primary single-view baseline and, to the best of our knowledge, represents the current state-of-the-art for parametric single-view relighting. As LightLab* shares the backbone, data, light conditioning, and training budget with SyncLight, differing only in the multi-view attention and LBM, it acts as a controlled ablation rather than a reproduction of the original model's numbers; its reference-view performance, within 2 dB of SyncLight on all splits, indicates a competitive implementation. For relighting the second view, we adopt LumiNet [62] (LNet), using the second view as input and the relit version of the reference view as the reference. We also compare against GR3EN [63], which addresses multi-view relighting. LuxRemix code and models are not available, so we use its synthetic data as an external benchmark instead (see tab. 2).

We additionally compare against Flux.2-dev [36], a large-scale text-conditioned image editing model. Since it lacks native multi-view support, we implement a two-stage pipeline: we use Qwen-2.5-7B-

[†]https://www.blenderkit.com/

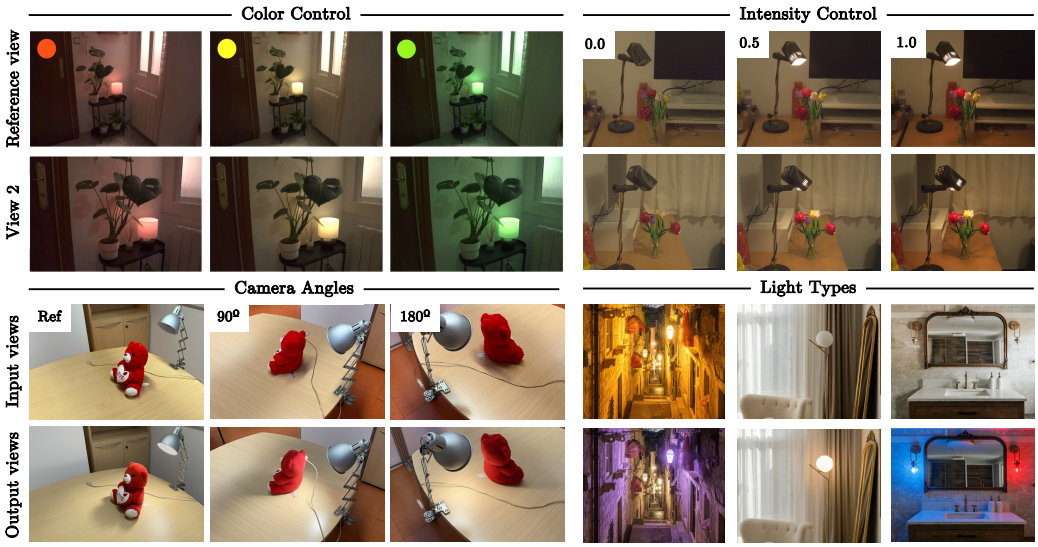


Figure 3: Qualitative results showcasing SyncLight's controllability and generalization. **Top-left:** precise control over light color, varying the chromaticity of a table lamp across three target hues. **Top-right:** continuous control over light intensity, from off (0.0) to fully on (1.0). **Bottom-left:** robustness to large camera angle changes, with input views rotated by up to 180° relative to the reference. **Bottom-right:** generalization across diverse scenes and unseen light types. SyncLight produces consistent relighting across all viewpoints.

Instruct [4] to generate a text prompt describing the desired lighting change from the source and target reference views, apply Flux.2-dev to relight the reference view, and then relight the second view by conditioning Flux.2-dev on the relit reference.

As shown in tab. 1 ("Reference (Ref.) view" and "Second (Sec.) view"), our method outperforms previous approaches by a large margin both on the reference and second views. Several key observations emerge from these results. First, SyncLight improves upon Lightlab* on the reference view, indicating that incorporating a second view enhances performance even for single-view relighting. Second, the results for both views in SyncLight are extremely close, clearly indicating the method's ability to consistently match the illumination conditions imposed for the first view with those in the second view. Finally, the average inference time across all images in our dataset demonstrates that our method achieves the most efficient relighting for both views.

Cross-view consistency. Per-image metrics do not measure cross-view agreement. A model can score well on each view while producing geometrically incompatible views. We therefore introduce VGGTm, the number of high-confidence VGGT [61] correspondences between the two output views, relative to that between the two ground-truth views (details in appendix D). A ratio near 100% means the outputs are as mutually 3D-consistent as the ground truth. On the Real split, SyncLight retains 95.8% of the ground-truth correspondences on the second view versus 82.7% for the best baseline, GR3EN (tab. 1), confirming genuine cross-view consistency rather than independently plausible per-view outputs.

Additional benchmark. To evaluate on data we did not build, we use the OLAT data of the LuxRemix [42] synthetic dataset to build an external test set of 500 view pairs with relit ground truth, unseen by all methods. SyncLight outperforms all baselines (tab. 2). It slightly improves on LightLab* on the reference view and, on the second view, gains 5.1 dB PSNR over GR3EN and 11.1 dB over LightLab*+LumiNet, the same trend as on our test set.

Scaling to many views. To further test the arbitrary-N claim, we render 32 views of each BlenderKit test scene and evaluate $N \in \{2, 4, 8, 16, 32\}$ (tab. 3): quality and consistency remain stable. With memory-efficient attention [11], the $NT \times NT$ attention matrix is never materialized, so memory grows only linearly (15.4 to 20.9 GB); compute grows quadratically but remains reasonable (43.7 s for 32 views; on a 24 GB RTX 4090, up to $N = 96$ views at 2K in 105 s). The Real split behaves consistently up to $N = 7$ (tab. 9 and fig. 17). In this dense setting, which favors video models, GR3EN still lags behind SyncLight while being over $3\times$ slower and using twice the memory.



Figure 4: Results on the Real Split from our dataset. SyncLight relights the images correctly, while the other methods struggle with consistency and overall quality.



Figure 5: Results on the benchmark built from the LuxRemix dataset. We compare Flux.2-dev, GR3EN, and SyncLight. In both examples, the goal is to turn on the light in white at maximum intensity.

Table 2: Additional benchmark built from the LuxRemix [42] dataset (500 unseen view pairs).

	Method	PSNR	SSIM	ΔE_{00}	VGGTm
Ref. view	Flux.2-dev	18.67	.662	15.84	-
	LightLab*	25.76	.800	7.18	-
	GR3EN	22.01	.752	10.43	-
	SyncLight	25.82	.803	7.16	-
Sec. view	Flux.2-dev	15.84	.656	20.83	48.2
	LightLab*+LNet	14.88	.682	21.92	53.8
	GR3EN	20.86	.738	11.92	82.1
	SyncLight	25.97	.808	7.02	90.6

Table 3: Scaling with the number of views N (BlenderKit, 32 views per scene). Time (s) and peak VRAM (GB) on an A40. Top: GR3EN with $N = 32$.

	N	PSNR	SSIM	ΔE_{00}	Time	VRAM
GR3EN		24.94	.896	8.25	$144_{\pm 4.21}$	40.8
SyncLight	2	28.14	.941	3.64	$1.58_{\pm .02}$	15.4
	4	27.92	.937	3.76	$2.17_{\pm .02}$	15.8
	8	27.79	.939	3.72	$5.82_{\pm .04}$	16.9
	16	28.09	.940	3.73	$17.39_{\pm .05}$	18.3
	32	27.97	.938	3.77	$43.71_{\pm .07}$	20.9

Qualitative results. SyncLight enables parametric control over light intensity and chromaticity, and generalizes to diverse scenes and light sources. Fig. 3 illustrates four scenarios. **Color Control** (top-left): SyncLight varies the chromaticity of a table lamp across three target hues at fixed intensity, with color bleed on the wall, plant, and floor propagating consistently to View 2. **Intensity Control** (top-right): smooth brightness modulation of a desk lamp from 0.0 to 1.0, with shadows cast by the flowers and vase sharpening coherently across both views. **Camera Angles** (bottom-left): given views rotated by approximately $90°$ and $180°$ around the scene, SyncLight produces coherent outputs in which the shadow direction rotates with the camera, suggesting that the model reasons about shadow geometry rather than copying lighting cues from nearby viewpoints. **Light Types** (bottom-right): generalization to diverse, light sources such as light poles, spherical lights, and vanity-style lights.

Fig. 4 shows qualitative results of competing methods and our SyncLight for the Real split of our dataset. As our dataset is color calibrated, we also present the GT on the last column. Results demonstrate our method's key capabilities: (1) Precise light manipulation: SyncLight correctly modifies individual light sources while preserving the remaining illumination, (2) Multi-view consistency: lighting edits propagate coherently across viewpoints, and (3) Indirect effects: SyncLight plausibly synthesizes indirect illumination effects such as wall reflections and color bleeding. In contrast, GR3EN struggles with the real scenes, while Flux.2-dev lacks cross-view consistency, producing results that look plausible in each view but are geometrically inconsistent across views. The LightLab*+LumiNet baseline, while maintaining better consistency by explicitly relighting each view, fails to propagate lighting effects. Fig. 5 shows two examples from the LuxRemix benchmark, each requiring a light to be turned on in pure white at maximum intensity. SyncLight produces the most consistent and realistic results.

Figure 6: Distribution of inter-view angles between reference and second view across our test split (red bars), overlaid with average PSNR per bin (blue curve).

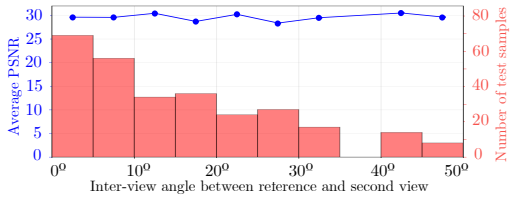


Table 4: Ablation studies. Average results on the "Second view" of the *Real* split of our dataset.

	PSNR	SSIM	ΔE_{00}	LPIPS
w/o *Infinigen*	26.91	.863	4.59	.281
w/o *BlenderKit*	27.29	.872	4.42	.247
w/o *Real*	25.61	.860	4.92	.238
w/o MV-SD	22.47	.833	5.03	.236
SyncLight	30.23	.897	3.56	.197

Inter-view angle performance analysis. A natural concern with multi-view relighting is whether performance degrades as the inter-view angle between the reference and the second view grows. To evaluate this, we bin the pairs of all test scenes by the angular displacement between the reference and second view, and report the average PSNR within each bin. Fig. 6 shows the resulting distribution (in red) and the average PSNR within each bin (blue curve). Importantly, the average PSNR remains essentially flat across the entire range, indicating that SyncLight maintains consistent relighting quality even for wide-baseline pairs.

Ablation studies. Tab. 4 presents ablation experiments. Dataset splits: Removing any split degrades performance on the real test split, with *Infinigen* (-3.3 dB in PSNR), *BlenderKit* (-2.9 dB in PSNR), and *Real* (-4.6 dB in PSNR). Multi-view transformer: Replacing multi-view blocks with standard single-view processing causes severe degradation on "Second view" (-7.7 dB in PSNR), reducing performance significantly. Without cross-view attention, the model cannot propagate lighting to other views, confirming that feature exchange across views is critical for multi-view consistency.

User study. We conducted a 20-participant user study evaluating relighting accuracy, multi-view consistency, and overall quality. Participants compared SyncLight, Flux.2-dev, and LightLab*, the latter provided with an additional lightmap for the second view. SyncLight was preferred in 73.9%, 81.1%, and 81.7% of cases, respectively, indicating superior perceptual quality and cross-view coherence. See supplementary material for details.

Zero-shot multi-view generalization. A crucial property of SyncLight is its ability to generalize zero-shot to an arbitrary number of views, despite being trained exclusively on image pairs ($N = 2$). This capability stems from our multi-view transformer architecture: because cross-view attention operates at the token level rather than being constrained to a fixed number of views, the learned consistency priors naturally extend to larger camera arrays. Fig. 7 presents an example with $N = 7$ views, demonstrating consistent relighting across all views despite exceeding the training configuration. Notably, the modified light source is occluded in views 2 and 4, yet the model correctly propagates the lighting changes based on geometric cues from other views. Moreover, at 2K resolution, each additional view requires only $\sim$0.15 GB VRAM and 1.23 seconds on an NVIDIA A40.

We quantitatively evaluate this generalization capability in tab. 1 (*Add. views*). For each image pair in our test set, we randomly sample additional views, obtaining an average of 4.38 views per scene. Baseline methods require a separate forward pass for each additional view, always using the first view as a reference. In contrast, SyncLight processes all views in a single forward pass. Our method maintains consistent relighting quality across the reference view, second view, and all additional views, demonstrating robust multi-view consistency beyond the training distribution.

Out-of-distribution scenes. We additionally evaluate on RealEstate10K [20] on fig. 8, a large-scale collection of indoor video sequences captured with moving cameras across diverse real estate scenes. Although RealEstate10K provides multiple views of the same scene, it lacks ground-truth relighting annotations, and we therefore use it for qualitative evaluation only. Fig. 8 shows that SyncLight generalizes to these unseen, in-the-wild scenes, producing consistent relighting across viewpoints despite the domain shift from our training data. In the scene the camera displacement is huge, and SyncLight is able to turn on the table lamp correctly. SyncLight is the only method that is able to perform the edits correctly. Note that our method also outperforms a per-view variant of LightLab*, where we relit each view independently, providing an additional lightmap for the second view.

Applications. Although trained on sparse view pairs, SyncLight also enables video relighting and 3D relighting with radiance fields (appendix H).



Figure 7: Relighting a scene with seven different views. Top row: Input images. Bottom row: SyncLight results. Note how all the views are consistently modified. We want to emphasize views 2 and 4, in which the lamp turned on is not visible, yet the effects in the scene match those of the other views.



Figure 8: Qualitative results on out-of-distribution images from the RealEstate10K dataset [20]. SyncLight outperforms all the other methods, including a per-view informed version of LightLab*, demonstrating that the joint processing of all views is beneficial even for single-light relighting.

6 Discussion

We presented SyncLight, a method for consistently controlling light sources across multiple uncalibrated views from a single reference edit, achieving strong qualitative and quantitative performance. SyncLight assumes (i) a static scene, (ii) synchronized captures, and (iii) that the light sources to be edited are visible in the chosen reference view. However, our work has limitations that suggest avenues for future research. First, SyncLight relies on a circular region for light conditioning and may struggle with rare or unconventional light sources; in particular, our work does not target windows. Finer-grained segmentation masks, as in LightLab [49], could improve generalization to such cases at the cost of increased interaction complexity. We deliberately avoided mask conditioning (as in LightLab, LuxRemix, and GR3EN) to keep SyncLight self-contained and efficient, without an external segmentation model. SyncLight can fail on very complex fixtures or in scenes with few or misleading visual cues. Second, leveraging priors from video models [41] could enable better generalization to videos, although such approaches may not transfer well to sparse multi-view sets with large baselines, such as those in RealEstate10K. Third, SyncLight changes the intensity and color of fixtures that already exist in the scene, but it is not designed to insert new light sources or to move existing ones. Combining our pose-free formulation with methods that offer CG-like light placement [8] is a promising way to remove this restriction. Our real split is also relatively small (40 scenes), and the large drop we observe when removing it suggests that more real multi-view captures would help close the remaining domain gap. Finally, incorporating 3D priors such as depth or camera pose may further improve performance and broaden generalization. We leave these directions to future work. Despite these limitations, SyncLight operates effectively on wide-baseline stereo setups and demonstrates zero-shot generalization across multiple cameras without relying on 3D information. We are confident that SyncLight marks the beginning of a broader research direction in multi-view, synchronous relighting. Each ingredient of SyncLight also extends naturally to outdoor scenes, windows and area lights, dynamic multi-camera video (our attention treats its input as a set, so all $N \times T$ latents can be processed jointly), and text-driven agentic interfaces; we outline these extensions in appendix C.

Broader impact statement Improved relighting models benefit content creation, AR, and VFX by enabling realistic lighting edits. The proposed method is a fundamental research contribution and does not have a direct negative societal impact; however, as with all image generation technologies, downstream applications must prioritize ethical use and guard against potential misuse.

Acknowledgements

This work was supported by Grant PID2024-162555OB-I00 funded by MCIN/AEI/10.13039/501100011033 and by ERDF, EU, AIA2025-163919-C52 funded by MCIN/AEI/10.13039/501100011033, the grant Catedra ENIA UAB-Cruïlla (TSI-100929-2023-2) from the Ministry of Economic Affairs and Digital Transformation of Spain, and by the Generalitat de Catalunya CERCA Program. DSL also acknowledges the FPI grant from Spanish Ministry of Science and Innovation (PRE2022-101525). JVC also acknowledges the Generalitat de Catalunya CERCA Program, and the 2025 Leonardo Grant for Scientific Research and Cultural Creation from the BBVA Foundation. The BBVA Foundation accepts no responsibility for the opinions, statements and contents included in the project and/or the results thereof, which are entirely the responsibility of the authors. Computing resources were partially provided by the Digital Research Alliance Canada.

References

[1] Yagiz Aksoy, Changil Kim, Petr Kellnhofer, Sylvain Paris, Mohamed Elgharib, Marc Pollefeys, and Wojciech Matusik. A dataset of flash and ambient illumination pairs from the crowd. In *Eur. Conf. Comput. Vis.*, 2018.

[2] Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *J. Mach. Learn. Research*, 26(209):1–80, 2025.

[3] Hadi Alzayer, Philipp Henzler, Jonathan T. Barron, Jia-Bin Huang, Pratul P. Srinivasan, and Dor Verbin. Generative multiview relighting for 3d reconstruction under extreme illumination variation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2025.

[4] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[5] Shrisha Bharadwaj, Haiwen Feng, Giorgio Becherini, Victoria Fernandez Abrevaya, and Michael J Black. Genlit: Reformulating single-image relighting as video generation. In *ACM SIGGRAPH Asia Conf.*, 2025.

[6] Anand Bhattad, Daniel McKee, Derek Hoiem, and David Forsyth. StyleGAN knows normal, depth, albedo, and more. In *Adv. Neural Inform. Process. Syst.*, 2023.

[7] Anand Bhattad, James Soole, and David A. Forsyth. Stylitgan: Image-based relighting via latent control. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2024.

[8] Chris Careaga and Yağız Aksoy. Physically controllable relighting of photographs. In *ACM SIGGRAPH Conf.*, 2025.

[9] Clément Chadebec, Onur Tasar, Sanjeev Sreetharan, and Benjamin Aubin. Lbm: Latent bridge matching for fast image-to-image translation. In *IEEE/CVF Int. Conf. Comput. Vis.*, 2025.

[10] Jun Myeong Choi, Annie Wang, Pieter Peers, Anand Bhattad, and Roni Sengupta. Scribblelight: Single image indoor relighting with scribbles. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2025.

[11] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[12] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of SIGGRAPH*, pages 145–156, 2000.

[13] Kangle Deng, Timothy Omernick, Alexander Weiss, Deva Ramanan, Jun-Yan Zhu, Tinghui Zhou, and Maneesh Agrawala. Flashtex: Fast relightable mesh texturing with lightcontrolnet. In *Eur. Conf. Comput. Vis.*, 2024.

[14] Xiaodan Du, Nicholas Kolkin, Greg Shakhnarovich, and Anand Bhattad. Generative models: What do they know? do they know things? let's find out! *arXiv preprint arXiv:2311.17137*, 2023.

[15] Yotam Erel, Rishabh Dabral, Vladislav Golyanik, Amit H. Bermano, and Christian Theobalt. Practilight: Practical light control using foundational diffusion models. *ACM Transactions on Graphics (TOG)*, 44(6):1–11, 2025.

[16] Ye Fang, Zeyi Sun, Shangzhan Zhang, Tong Wu, Yinghao Xu, Pan Zhang, Jiaqi Wang, Gordon Wetzstein, and Dahua Lin. Relightvid: Temporal-consistent diffusion model for video relighting. *arXiv preprint arXiv:2501.16330*, 2025.

[17] Frédéric Fortier-Chouinard, Zitian Zhang, Louis-Etienne Messier, Mathieu Garon, Anand Bhattad, and Jean-François Lalonde. Spotlight: Shadow-guided object relighting via diffusion. In *Int. Conf. 3D Vis.*, 2026.

[18] James Gardner, Evgenii Kashin, Bernhard Egger, and William Alfred Peter Smith. The sky's the limit: Relightable outdoor scenes via a sky-pixel constrained illumination prior and outside-in visibility. In *Eur. Conf. Comput. Vis.*, 2024.

[19] Marc-André Gardner, Yannick Hold-Geoffroy, Kalyan Sunkavalli, Christian Gagné, and Jean-François Lalonde. Deep parametric indoor lighting estimation. In *IEEE/CVF Int. Conf. Comput. Vis.*, 2019.

[20] Google Research. RealEstate10K: A large-scale dataset of camera poses. `https://google.github.io/realestate10k/`, 2018.

[21] Paul Haeberli. Synthetic lighting for photography. Grafica Obscura, `http://www.graficaobscura.com/synth/`, 1992.

[22] Kai He, Ruofan Liang, Jacob Munkberg, Jon Hasselgren, Nandita Vijaykumar, Alexander Keller, Sanja Fidler, Igor Gilitschenski, Zan Gojcic, and Zian Wang. Unirelight: Learning joint decomposition and synthesis for video relighting. In *Adv. Neural Inform. Process. Syst.*, 2025.

[23] Mingming He, Pascal Clausen, Ahmet Levent Taşel, Li Ma, Oliver Pilarski, Wenqi Xian, Laszlo Rikker, Xueming Yu, Ryan Burgert, Ning Yu, et al. Diffrelight: Diffusion-based facial performance relighting. In *ACM SIGGRAPH Asia Conf.*, 2024.

[24] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. Deep outdoor illumination estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[25] Zhuo Hui, Kalyan Sunkavalli, Sunil Hadap, and Aswin C Sankaranarayanan. Illuminant spectra-based source separation using flash photography. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2018.

[26] Haian Jin, Yuan Li, Fujun Luan, Yuanbo Xiangli, Sai Bi, Kai Zhang, Zexiang Xu, Jin Sun, and Noah Snavely. Neural gaffer: Relighting any object via diffusion. In *Adv. Neural Inform. Process. Syst.*, 2024.

[27] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensoir: Tensorial inverse rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2023.

[28] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *ACM Trans. Graph.*, 2011.

[29] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. Automatic scene inference for 3d object compositing. *ACM Trans. Graph.*, 2014.

[30] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2024.

[31] Bingxin Ke, Kevin Qu, Tianfu Wang, Nando Metzger, Shengyu Huang, Bo Li, Anton Obukhov, and Konrad Schindler. Marigold: Affordable adaptation of diffusion-based image generators for image analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1–18, 2025.

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

[33] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[34] Peter Kocsis, Julien Philip, Kalyan Sunkavalli, Matthias Nießner, and Yannick Hold-Geoffroy. Lightit: Illumination modeling and control for diffusion models. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2024.

[35] Peter Kocsis, Vincent Sitzmann, and Matthias Nießner. Intrinsic image diffusion for single-view material estimation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2024.

[36] Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

[37] Junxuan Li, Hongdong Li, and Yasuyuki Matsushita. Lighting, reflectance and geometry estimation from 360° panoramic stereo. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2021.

[38] Zhen Li, Lingli Wang, Mofang Cheng, Cihui Pan, and Jiaqi Yang. Multi-view inverse rendering for large-scale real-world indoor scenes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2023.

[39] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2020.

[40] Zhengqin Li, Jia Shi, Sai Bi, Rui Zhu, Kalyan Sunkavalli, Miloš Hašan, Zexiang Xu, Ravi Ramamoorthi, and Manmohan Chandraker. Physically-based editing of indoor scene lighting from a single image. In *Eur. Conf. Comput. Vis.*, 2022.

[41] Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Zhi-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, and Zian Wang. Diffusionrenderer: Neural inverse and forward rendering with video diffusion models. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, June 2025.

[42] Ruofan Liang, Norman Müller, Ethan Weber, Duncan Zauss, Nandita Vijaykumar, Peter Kontschieder, and Christian Richardt. Luxremix: Lighting decomposition and remixing for indoor scenes. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2026.

[43] Zhi-Hao Lin, Bohan Liu, Yi-Ting Chen, Kuan-Sheng Chen, David Forsyth, Jia-Bin Huang, Anand Bhattad, and Shenlong Wang. Urbanir: Large-scale urban scene inverse rendering from a single video. In *Int. Conf. 3D Vis.*, 2025.

[44] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *Int. Conf. Learn. Represent.*, 2023.

[45] Yehonathan Litman, Fernando De la Torre, and Shubham Tulsiani. Lightswitch: Multi-view relighting with material-guided diffusion. In *IEEE/CVF Int. Conf. Comput. Vis.*, 2025.

[46] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *Int. Conf. Learn. Represent.*, 2023.

[47] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2024.

[48] Linjie Lyu, Valentin Deschaintre, Yannick Hold-Geoffroy, Miloš Hašan, Jae Shin Yoon, Thomas Leimküehler, Christian Theobalt, and Iliyan Georgiev. Intrinsicedit: Precise generative image manipulation in intrinsic space. *ACM Trans. Graph.*, 44(4), 2025.

[49] Nadav Magar, Amir Hertz, Eric Tabellion, Yael Pritch, Alex Rav-Acha, Ariel Shamir, and Yedid Hoshen. Lightlab: Controlling light sources in images with diffusion models. In *ACM SIGGRAPH Conf.*, 2025.

[50] Yiqun Mei, Mingming He, Li Ma, Julien Philip, Wenqi Xian, David M George, Xueming Yu, Gabriel Dedic, Ahmet Levent Taşel, Ning Yu, et al. Lux post facto: Learning portrait performance relighting with conditional video diffusion and a hybrid dataset. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2025.

[51] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[52] Lukas Murmann, Michael Gharbi, Miika Aittala, and Fredo Durand. A dataset of multi-illumination images in the wild. In *IEEE/CVF Int. Conf. Comput. Vis.*, 2019.

[53] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *Int. Conf. Learn. Represent.*, 2024.

[54] Yohan Poirier-Ginter, Alban Gauthier, Julien Phillip, J-F Lalonde, and George Drettakis. A diffusion approach to radiance field relighting using multi-illumination synthesis. *Comput. Graph. Forum*, 43(4), 2024.

[55] Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, et al. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2024.

[56] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2022.

[57] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. MVDream: Multi-view diffusion for 3d generation. In *Int. Conf. Learn. Represent.*, 2024.

[58] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion Schrödinger bridge matching. In *Adv. Neural Inform. Process. Syst.*, 2023.

[59] Zhenxiong Tan, Zeqing Wang, Xingyi Yang, Songhua Liu, and Xinchao Wang. Vision bridge transformer at scale. *arXiv preprint arXiv:2511.23199*, 2025.

[60] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH Conf.*, 2023.

[61] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[62] Xiaoyan Xing, Konrad Groh, Sezer Karagolu, Theo Gevers, and Anand Bhattad. Luminet: Latent intrinsics meets diffusion models for indoor scene relighting. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2025.

[63] Xiaoyan Xing, Philipp Henzler, Junhwa Hur, Runze Li, Jonathan T Barron, Pratul P Srinivasan, and Dor Verbin. Gr3en: Generative relighting for 3d environments. In *ACM SIGGRAPH Conf.*, 2026.

[64] Zongsheng Yue, Jianyi Wang, and Chen Change Loy. Resshift: Efficient diffusion model for image super-resolution by residual shifting. In *Adv. Neural Inform. Process. Syst.*, 2023.

[65] Chong Zeng, Yue Dong, Pieter Peers, Youkang Kong, Hongzhi Wu, and Xin Tong. Dilightnet: Fine-grained lighting control for diffusion-based image generation. In *ACM SIGGRAPH Conf.*, 2024.

[66] 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 Conf.*, 2024.

[67] Edward Zhang, Michael F. Cohen, and Brian Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Trans. Graph.*, 35(6), 2016.

[68] 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 *Int. Conf. Learn. Represent.*, 2025.

[69] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2018.

[70] Xiao Zhang, William Gao, Seemandhar Jain, Michael Maire, David Forsyth, and Anand Bhattad. Latent intrinsics emerge from training to relight. In *Adv. Neural Inform. Process. Syst.*, 2024.

[71] Zitian Zhang, Frédéric Fortier-Chouinard, Mathieu Garon, Anand Bhattad, and Jean-François Lalonde. Zerocomp: Zero-shot object compositing from image intrinsics via diffusion. In *IEEE/CVF Winter Conf. App. Comput. Vis.*, 2025.

[72] Zitian Zhang, Iliyan Georgiev, Michael Fischer, Yannick Hold-Geoffroy, Jean-François Lalonde, and Valentin Deschaintre. UniLight: A unified representation for lighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[73] Xiaoming Zhao, Pratul P. Srinivasan, Dor Verbin, Keunhong Park, Ricardo Martin Brualla, and Philipp Henzler. Illuminerf: 3d relighting without inverse rendering. In *Adv. Neural Inform. Process. Syst.*, 2024.

[74] Jingsen Zhu, Yuchi Huo, Qi Ye, Fujun Luan, Jifan Li, Dianbing Xi, Lisha Wang, Rui Tang, Wei Hua, Hujun Bao, and Rui Wang. I^2-sdf: Intrinsic indoor scene reconstruction and editing via raytracing in neural sdfs. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2023.

A Dataset details

Comparison with existing datasets. Existing datasets fall short of the three criteria required for multi-view relighting (diverse indoor geometry, synchronised multi-view captures, and isolated controllable light sources). The Multi-Illumination dataset [52], while offering real-world lighting variations, does not contain consistent multi-view captures or visible light sources, preventing its use for multi-view propagation. The dataset used by LightLab [49] is single-view only and is not publicly available. These limitations motivate the construction of the SyncLight dataset.

Light composition pipeline. We first obtain an ambient image I_{amb} with all controllable lights turned off, capturing only ambient illumination. Then, we render (or capture) the scene lit by each of the N light sources individually following the OLAT protocol, yielding a set of images $\{I_i\}$ for light $i \in \{1, ..., M\}$. By subtracting the ambient light from each OLAT capture, we obtain the isolated contribution of each light source: $L_i = I_i - I_{\text{amb}}$. We can then generate arbitrary lighting combinations with different intensities and colors through weighted linear combinations:

$$I_{\text{relit}} = I_{\text{amb}} + \sum_{i=1}^{M} c_i L_i \,, \tag{5}$$

where $c_i \in \mathbb{R}^3$ specifies the desired color (and intensity) of light i. In practice, c_i is originally defined in Lab space as to more intuitively separate luminance from chromaticity and then converted to RGB before applying the combination. Finally, we apply tone mapping to convert the linear result to a non-linear 8-bit image. We use Reinhard method with a $\gamma = 2.2$.

Dataset sources. Our dataset presents images from three different sources: Infinigen, BlenderKit, and Real captures. Infinigen images enable the model to leverage pretrained scene-geometry priors across views. BlenderKit images contribute increased realism, including furniture and light sources. Finally, Real capture images help reduce the domain gap between synthetic and real-world data.

Infinigen. We utilize Infinigen [55], a procedural generator of photorealistic 3D Blender scenes, to create 365 diverse indoor environments across five room types: bathrooms, bedrooms, living rooms, dining rooms, and kitchens. Each scene exhibits high variation in geometry, materials, and assets. We procedurally place 5 camera groups, each comprising 5 cameras, pointing at different parts of the scene. We ensure at least 25% spatial overlap between all viewpoints within the same group to maintain multi-view consistency constraints. For each scene, we identify all emissive objects (e.g., lamps and ceiling lights) and render them following the OLAT protocol. Each view is rendered in HDR with physically-based rendering, producing a total of 920,000 pairs of different lighting combinations.

BlenderKit. To increase realism beyond procedurally-generated scenes, we curate 40 high-quality Blender scenes from BlenderKit[‡], a repository of artist-created 3D assets and scenes. These scenes feature more realistic furniture arrangements, diverse architectural styles, and complex lighting fixtures. To integrate these scenes into our pipeline, we perform several preprocessing steps: (i) we replace all emissive materials with transparent materials and explicit Light objects to enable precise OLAT control; (ii) we manually position 3–6 cameras per scene following the same multi-view overlap constraints as *Infinigen*; (iii) we remove unrealistic or low-quality assets; and (iv) we augment scenes with additional BlenderKit assets including diverse lamps, light fixtures, and decorative objects to enhance scene complexity and realism. Each view is rendered in HDR format, similarly to Infinigen, producing 47,000 pairs with diverse lighting combinations.

Real captures. To bridge the gap between synthetic and real-world imagery, we capture 320 RAW photographs from 40 indoor scenes. We use a DSLR camera (Canon EOS 1300D) and a mobile device (iPhone 13), fixed on a tripod and activated via a remote trigger. For each viewpoint, we fix the camera parameters (ISO, shutter speed, and aperture for the DSLR; ISO and shutter speed for the mobile device) to preserve photometric consistency across OLAT captures. Exposure settings are selected to minimize clipping under active lighting while keeping the ambient capture sufficiently bright to avoid excessive noise. We reposition the tripod to acquire multiple viewpoints per scene and capture each scene following the same multi-view OLAT protocol as in the synthetic splits. We then manually annotate light source locations in the images and apply the same light composition pipeline to generate new lighting conditions. This yields 18,000 real-image pairs with diverse lighting configurations.

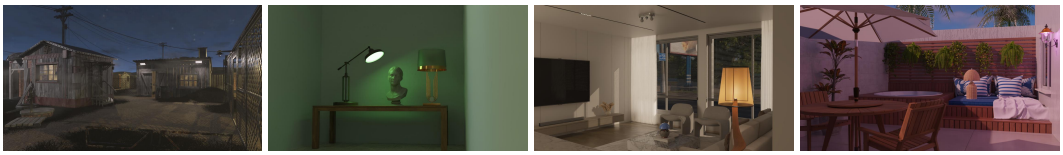
Test split. For testing, we hold out a diverse subset of scenes not seen during training: 10 from Infinigen, 3 from BlenderKit, and 6 from Real captures. From these unseen scenes, we randomly sample a fixed test set of 135, 45, and 105 pairs, respectively. This held-out split is used for all quantitative evaluations reported in the main paper.

Fig. 9 illustrates examples from the SyncLight dataset. Images and light conditions were selected randomly.

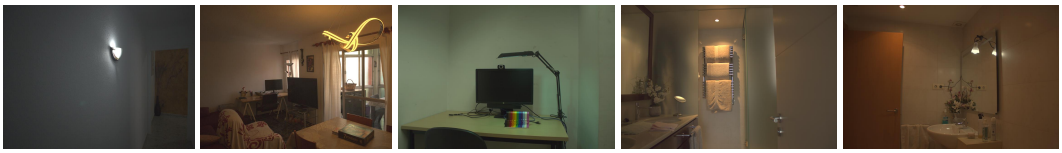
[‡]https://www.blenderkit.com/



(a) *Infinigen* split



(b) *BlenderKit* split



(c) *Real* split

Figure 9: Examples from the SyncLight dataset. Samples were selected randomly with random light colors.

B Further method analysis and details

Interpolant stochasticity. Flow matching interpolates between two samples through a deterministic linear combination. In contrast, bridge matching introduces a stochastic component via Gaussian noise:

$$\mathbf{z}_t = (1-t)\mathbf{z}_{src} + t\mathbf{z}_{tar} + \sigma\sqrt{t(1-t)}\boldsymbol{\epsilon}\,, \tag{6}$$

where $\mathbf{z}_{\text{src}}$ and $\mathbf{z}_{\text{tar}}$ are (latent) samples from the source and target distributions, respectively, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. The parameter σ controls the level of stochasticity in the interpolant. Tab. 5 evaluates four different values of σ. For each row, we retrain SyncLight using four steps ($T = 4$), but with different values of σ. We find that $\sigma = 0.005$ achieves optimal performance, while both deterministic flow matching ($\sigma = 0$) and higher noise levels ($\sigma > 0.005$) result in degraded performance.

Number of timesteps T during training. We employ four timesteps during training. It is crucial to learn a well-conditioned vector field rather than a brittle point-to-point mapping. As the broader flow matching literature shows [44], training over multiple timesteps yields a generalizable transport, even if inference is truncated to fewer steps [9]. Reducing the number of timesteps to one ($T_{train} = 1$) during training reduces the formulation to an L_2 loss. To validate this, we trained SyncLight using only one timestep during training (using $t = 0$ in eq. (6)). Tab. 6 reports a 4.82dB drop in PSNR on the Real split of our dataset compared to our $T_{train} = 4$ formulation. Multi-step training provides a clear, measurable benefit.

Number of timesteps at inference time. SyncLight is trained with four equally spaced timesteps (t in eq. (6)). Surprisingly, we find that using only a single timestep during inference is both faster and more accurate, as shown in tab. 7. Note that for this experiment, the model weights are the ones obtained using four timesteps during training, and even with that, one-step inference gives better performance. We attribute this counterintuitive result to the initialization strategy. Unlike standard text-to-image models that start from Gaussian noise, our framework begins with the input images as $\mathbf{z}_{\text{src}}$, which already provides a strong initialization. Under this regime, each additional timestep introduces a discretization error that accumulates across the trajectory, ultimately degrading the quality of the final output.

Velocity estimator discussion. We employ Stable Diffusion as the velocity estimator in our bridge matching formulation. While video diffusion models may appear suitable for multi-image processing, their architectures are designed for temporally coherent sequences where consecutive frames exhibit high visual similarity. Our multi-view scenario differs fundamentally: synchronized views from different camera positions contain substantially different visual content despite depicting the same scene. The temporal-encoding mechanisms in video VAEs, optimized for frame-to-frame coherence, struggle to capture this sparse multi-view setting. Instead, we leverage image diffusion models and introduce explicit multi-view reasoning through our transformer-based attention

Table 5: Effect of the noise level σ.

	PSNR	SSIM	ΔE_{00}	LPIPS
$\sigma = 0$ (FM)	29.71	.883	3.91	.203
$\sigma = 0.001$	30.03	.884	3.87	.210
$\sigma = 0.005$	30.29	.896	3.65	.196
$\sigma = 0.01$	26.48	.863	4.76	.228

Table 6: Effect of the number of steps during training T. For inference, we use only one timestep on both cases.

	PSNR	SSIM	ΔE_{00}	LPIPS
$T_{train} = 1$	25.47	.858	5.02	.243
$T_{train} = 4$	30.29	.896	3.65	.196

Table 7: Effect of the number of inference steps.

	PSNR	SSIM	ΔE_{00}	LPIPS
One-step	30.29	.896	3.65	.196
Two-step	30.04	.893	3.71	.197
Four-step	29.98	.893	3.74	.200

Table 8: Effect of the color space used for the LBM formulation.

	PSNR	SSIM	ΔE_{00}	LPIPS
RGB	30.09	.894	3.82	.199
CIE Lab	30.29	.896	3.65	.196

mechanism. We also experimented with Flux-based DiT backbones early on, but they failed to propagate the relighting to the additional views.

Multi-view transformer block further details. SyncLight takes as input two synchronized, uncalibrated views under source lighting, together with a target lighting condition specified on the reference view, and predicts both views under the target lighting. Standard diffusion- and flow-based architectures are designed for single-image inputs and do not model cross-view interactions. We therefore modify the transformer blocks of Stable Diffusion XL to operate in our setting while preserving its pretrained weights. In contrast to text-to-3D methods such as MVDream [57] and Wonder3D [47], our approach does not require camera poses, as we aim at synchronized light source editing rather than 3D generation. The key idea is to enable information exchange across N views through a modified self-attention mechanism (see fig. 10 for an illustration with $N = 2$). Given a batch of N view latents, we first process each view's features independently through the initial convolutional layers. Before each transformer block, features from all views are concatenated along the token dimension rather than the batch dimension, allowing features from different views to interact within the multi-view transformer block. Fig. 11 shows an example with $N = 3$ views, where a table lamp is turned on with a red color and the angle between the reference view and one of the other views reaches up to 90°

Lightmap codification. Finally, we analyze the choice of color space for lightmap encoding. We employ the CIE Lab color space to encode target light colors, as it provides a perceptually uniform color representation. Tab. 8 compares Lab encoding against RGB. While performance is similar, Lab achieves marginally better results and offers user-friendliness. Its inherent separation of luminance (L) and chromaticity (ab) enables independent control over light intensity and color, which is more intuitive for users.

The effect of diverse domains Fig. 12 complements the Ablation study from the main paper, showing qualitative results from our model when each of the dataset splits is removed at a time. In detail, we can see how lamps can lose realism when either the *Real captures* or the *BlenderKit* splits are not used—see the dark color of the ceiling lamp in the bottom figure. In contrast, not using the large number of images from the *Infinigen* split forces the model to output slightly brighter results.

Implementation details. We fine-tune our modified SDXL model [53] for 100,000 steps using a learning rate of 1×10^{-5} and a batch size of 6 at a resolution of 1280×720. Training is performed on six NVIDIA A40 GPUs and takes 8 days. $\lambda = 10$ in all experiments. Although *Infinigen* dominates the pair count ($\sim$920K of $\sim$985K pairs), during training we sample uniformly from the three splits, so the *Real* and *BlenderKit* splits are effectively upweighted. The splits play complementary rather than interchangeable roles: *Infinigen* provides cheap geometric diversity that helps preserve the pretrained prior, while the other two provide realism (tab. 4).

C Extensions and future directions

Each ingredient of SyncLight generalizes independently. *Outdoor scenes* could be handled by conditioning on sun direction and strength [24], on encoded environment maps [22], or on a lighting token [72]. *Windows and area lights* only require replacing circular markers with segmentation masks in the same 4-channel Lab encoding, a drop-in change to the conditioning rather than to the architecture. *Dynamic scenes* need no architectural change either: our multi-view attention treats its input as a set, so the $N \times T$ latents of a multi-camera video can be processed as one set of tokens while the user still edits a single frame; training only requires adding temporal

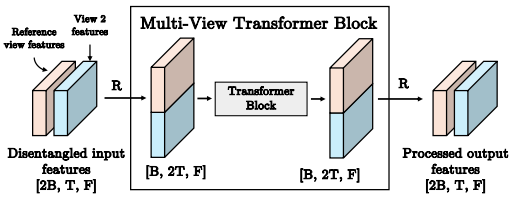


Figure 10: Multi-view transformer block. To ensure consistent relighting across viewpoints, we modify the standard self-attention mechanism of SDXL. View features are concatenated and reshaped along the sequence dimension, creating a unified representation of shape $[B, 2T, F]$



Figure 11: Example of SyncLight with $N = 3$ views. Despite up to $\sim 90^\circ$ between the reference view and the farthest view, SyncLight produces consistent relighting across all viewpoints when turning on a red table lamp.



Figure 12: Multi-view relighting examples when training SyncLight without one of the splits of the dataset at a time.

pairs to the batch. Finally, an *agentic interface* could remove manual interaction: a VLM parses a text prompt, a segmentation model such as SAM [33] localizes the named source in the reference view, and pseudo-Lab values are derived from the color name, making text and clicks two entry points to the same conditioning, with text trading some color and intensity precision for convenience.

D Cross-view consistency metric (VGGTm)

Per-image metrics (PSNR, SSIM, ΔE_{00}, LPIPS) compare each output with its own ground truth and therefore cannot tell whether two outputs agree with each other: a model can score well on every view while producing geometrically incompatible views. VGGTm addresses this with VGGT [61]. For each test scene, we extract correspondences between the two ground-truth views and, separately, between the two output views, and count the high-confidence matches in each case. VGGTm is the ratio between the output count and the ground-truth count, expressed as a percentage. A ratio near 100% indicates that the relit views remain as mutually 3D-consistent as the ground truth, whereas a lower ratio reveals cross-view hallucinations, such as shadows or highlights that do not agree across views, that pixel-wise metrics miss.

Because VGGT's correspondence confidence is scene-dependent (it degrades on wide-baseline or low-overlap pairs regardless of relighting quality), we gate applicability on the ground-truth pair's own confidence, which is

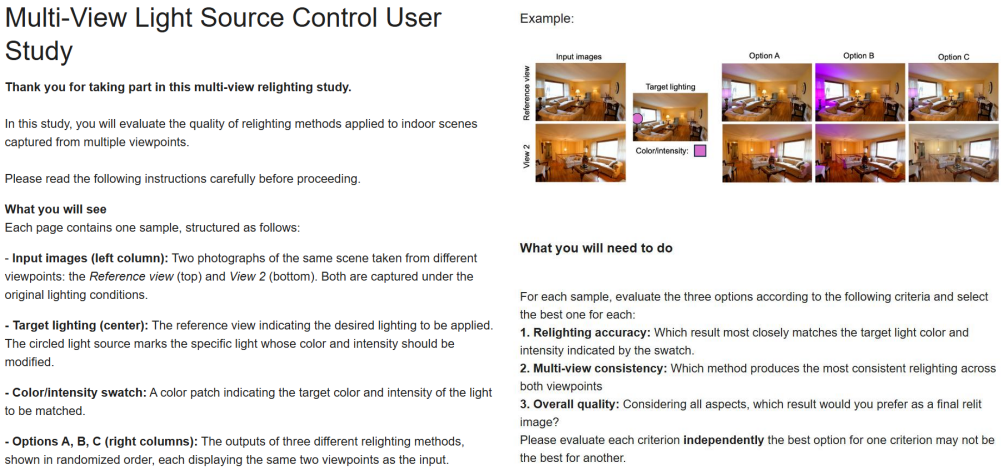


Figure 13: Instructions given to the users for the User Study.

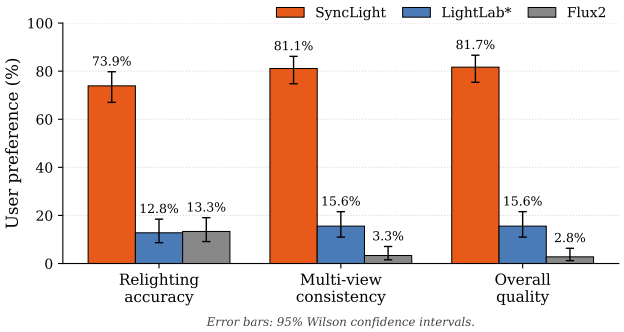


Figure 14: User study results.

independent of the model under evaluation, and report the metric only on scenes that clear this bar. For additional views, the metric is averaged over view pairs.

E Scalability analysis

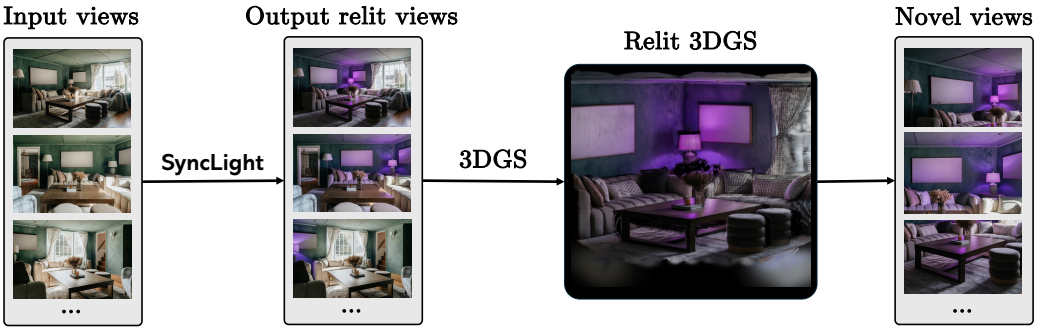
Fig. 17 plots inference time and peak memory against the number of views N for the 32-view BlenderKit experiment of tab. 3 in the main paper. We use memory-efficient attention (FlashAttention [11] through PyTorch SDPA), so the $NT \times NT$ attention matrix is never materialized and memory scales as $\mathcal{O}(N \cdot T)$: it grows linearly, from 15.4 GB at $N = 2$ to 20.9 GB at $N = 32$. Compute grows quadratically with N, as expected for full cross-view attention, yet remains practical. Tab. 9 repeats the analysis on the Real test split for $N \in \{2, 4, 6, 7\}$: performance is consistent regardless of the number of views. Beyond these ranges, the applications in appendix H push N much further: a 142-frame video is relit jointly in 156 s, and radiance-field reconstruction from relit views remains consistent.

F Local versus global illumination changes

Editing a light source can change the overall ambience of the scene. This is physically expected rather than a degradation: toggling or recoloring a light changes the indirect illumination throughout the scene, and the ground truth exhibits the same shift. Our targets are formed by OLAT composition in linear space (HDR renders, camera RAW), $I_{\text{relit}} = I_{\text{amb}} + \sum_i c_i L_i$, the same procedure used by LightLab [49], LuxRemix [42], and GR3EN [63], with roots in light stage capture [12]. Because the combination is linear, the ground truth is photometrically exact: the ambient term I_{amb} is preserved, and the relative size of the ambience shift is set by the ratio between the edited source and the illumination already present. A scene lit by strong daylight therefore changes little when a dim source is turned on.



(a) Video relighting. From user inputs in the first frame only, SyncLight automatically relights 142 frames of a video from the RealEstate10k dataset [20].



(b) Novel-view synthesis. 3DGS [32] is applied to images relit by SyncLight, which achieves out-of-the-box multi-view consistency.

Figure 15: SyncLight applied to: (a) video relighting, and (b) novel-view synthesis with radiance fields. Please see the supplementary video on our webpage.

Table 9: Scaling with the number of views N on the Real test split.

N	PSNR	SSIM	ΔE_{00}	LPIPS	VGGTm	Inf. time (s)
2	30.23	.897	3.56	.197	95.8	1.58 ± 0.02
4	30.40	.904	3.55	.194	94.8	2.17 ± 0.02
6	30.45	.907	3.56	.194	94.9	3.01 ± 0.03
7	30.47	.906	3.56	.192	95.3	3.83 ± 0.03

To verify that SyncLight does not shift the ambience more than the ground truth, tab. 11 separates weak from strong edits, measured by the change in requested lightness $|\Delta L|$, and groups pixels into rings around the edited source at increasing distance, in multiples of the marker radius r. The first two metric columns compare our output with the ground truth; the last two compare the ground truth and our output with the input image, measuring how much the scene changed in the first place. For weak edits, pixels far from the edited light stay close to the input: the ground truth changes by $0.9\,\Delta E_{00}$ there and our output by 1.0, so the ambience is nearly unchanged and SyncLight does not shift it more than the ground truth does. For strong edits, both numbers are much larger, as the far region covers most of the image and absorbs the indirect illumination. In both cases, our output tracks the ground truth, so the size of the ambience shift is set by the scene and the edit, not by the model.

G User study

Standard image-quality metrics such as PSNR, SSIM, ΔE_{00}, and LPIPS measure pixel- or feature-level distance to a reference image, but do not capture key perceptual aspects of relighting: fidelity to the requested light colour and intensity, plausibility of shadows and inter-reflections, and coherence across viewpoints. We therefore complement our quantitative evaluation with a user study assessing these perceptual qualities directly.

We present the instructions given to participants in fig. 13. Participants were first introduced to the overall layout of the experimental interface through both written instructions and a visual example, and were then given explanations of the three types of questions they would be asked. For each sample, participants were shown a reference view together with a target lighting specification and the corresponding outputs from three

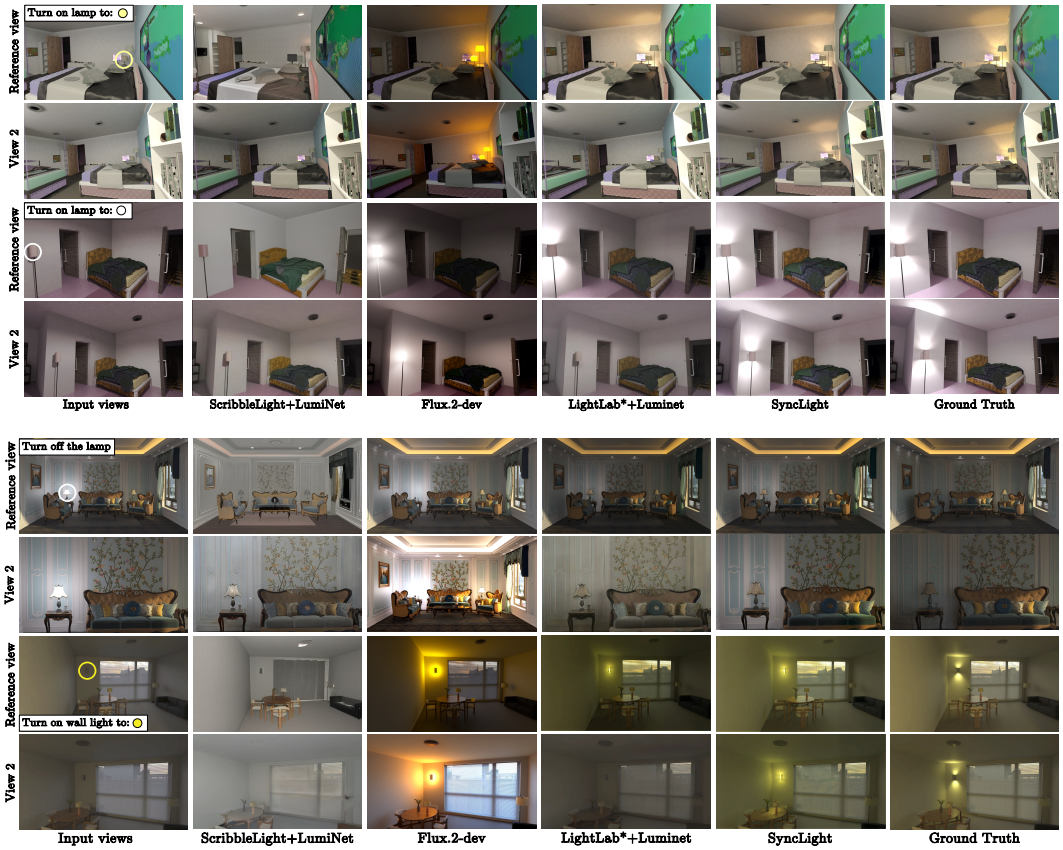


Figure 16: Qualitative results on the SyncLight dataset on the *Infinigen* (1–2), and *BlenderKit* (3–4) splits. *Real* split is shown in the main paper.

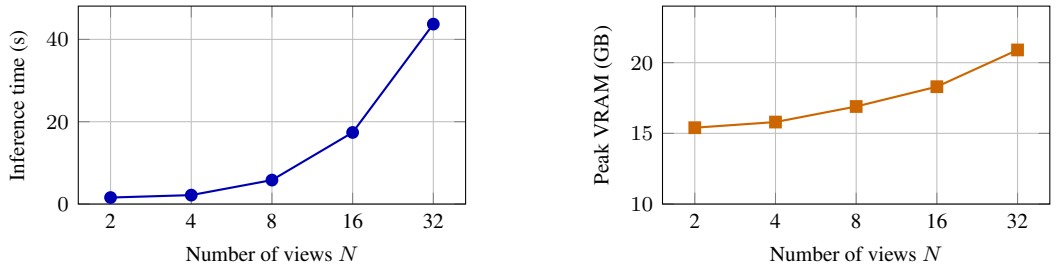


Figure 17: Inference time (left) and peak memory (right) of SyncLight as a function of the number of views N on an NVIDIA A40 (BlenderKit, 32 rendered views per scene). Memory grows linearly; compute grows quadratically.

methods: SyncLight, LightLab* (provided with a per-view lightmap) [49], and Flux2 [36]. Although our method internally uses CIE Lab, we present the target lighting to participants as an RGB swatch encoding the desired color and intensity, for simplicity. The light source to be manipulated is indicated by a circle marker overlaid on the reference view. The order of the three methods was randomized per sample to eliminate position bias. Participants were asked to answer three questions, each targeting a distinct perceptual axis of relighting quality:

1. **Relighting accuracy:** Which output is closest to the target light color and intensity? This question evaluates fidelity to the user-specified edit, independent of considerations of consistency or realism.
2. **Multi-view consistency:** Which output exhibits the most consistent lighting between the two views (e.g., coherent shadow direction, matching color casts, agreement on light source state)? This question isolates the cross-view coherence that distinguishes SyncLight from per-view baselines.
3. **Overall quality:** Which output is the most visually convincing overall? This question captures the holistic perceptual judgement that integrates accuracy, consistency, and the absence of artefacts.

Table 10: User input required by each method.

Method	Interaction per edited light	Cost per additional view
SyncLight	one click + color + intensity	none
LightLab [49]	segmentation mask + color + intensity	one mask per view
GR3EN [63]	segmentation mask + color + intensity	one mask per view + camera poses
LuxRemix [42]	OLAT harmonization + color + intensity	Plücker embeddings + harmonization

Table 11: Local versus global changes on the Real split. Weak and strong edits are separated by the requested lightness change $|\Delta L|$; pixels are grouped by their distance to the edited source in multiples of the marker radius r. The last row reports all edits over the full image, as in tab. 1.

	Ours vs. GT		ΔE_{00} vs. input	
Region	PSNR	ΔE_{00}	GT	Ours
Weak edits ($\|\Delta L\| < 0.5$)				
0–1 r (marker)	30.6	3.12	7.9	8.2
1–2 r	30.9	2.84	3.1	3.3
2–4 r	31.8	2.62	1.6	1.7
$>4\,r$	32.1	2.71	0.9	1.0
Full image	31.7	2.83	1.4	1.5
Strong edits ($\|\Delta L\| > 0.5$)				
0–1 r (marker)	33.1	6.31	23.4	23.9
1–2 r	32.8	4.78	12.7	13.1
2–4 r	31.0	4.26	7.4	7.6
$>4\,r$	29.2	4.08	5.8	6.0
Full image	29.9	4.21	7.1	7.3
All edits, full image	30.2	3.56	–	–

We collected responses from 20 participants, each answering all three questions for 10 samples, yielding 200 votes per question. Fig. 14 summarizes the results. SyncLight is preferred across all three criteria by 73.9% (relighting accuracy), 81.1% (multi-view consistency), and 81.7% (overall quality), with non-overlapping 95% Wilson confidence intervals between SyncLight and the next-best method in every case. Notably, and in line with our motivation and the results in tab. 1, SyncLight outperforms LightLab* in all three criteria despite the latter being provided with a per-view lightmap as additional input. This indicates that jointly processing all views improves not only multi-view consistency but also single-view relighting accuracy, since the model can leverage cross-view evidence (e.g., shading cues visible in one view but not in another) to better estimate the target lighting in each frame. Flux2 achieves results comparable to LightLab* on relighting accuracy, which we attribute to its strength as a general-purpose image editor capable of producing plausible global colour and contrast adjustments. However, as the consistency and overall quality results show, this comes at the cost of geometric coherence across views and frequently introduces hallucinations or physically implausible artifacts.

H Applications

SyncLight is trained on pairs of views $N = 2$. However, in the main manuscript, we show how our approach generalizes zero-shot to multiple views $N > 2$. We always aim at sparse view scenarios where the synchronous views are have a displacement between them. Having said this, there are two applications in which SyncLight is useful: video relighting and 3D relighting. See the attached video to visualize the results for both applications.

Video relighting SyncLight can efficiently relight videos while maintaining temporal consistency, even if it is trained only on *image pairs*. Note that the user only needs to specify the desired lighting condition on the first frame of the video, and SyncLight propagates the light changes across all the following frames. Fig. 15a shows five frames from a 142-frame video from the RealEstate10k dataset [20] alongside their relit versions, demonstrating consistent relighting across all frames. Notably, all 142 frames are processed jointly in a single forward pass, requiring only 156 seconds and 27 GB of VRAM.

Novel view synthesis with radiance fields Relighting in radiance field representations (NeRFs, or 3DGS) is inherently challenging since it typically involves complex inverse rendering (e.g., [27]). SyncLight's view-consistent relighting enables an alternative approach: users can relight multiple 2D views and reconstruct the 3D scene, effectively using our method as a proxy for 3D relighting. Fig. 15b and the supplementary video show an example using 3DGS [32] implementation from NerfStudio [60]. Note that, as opposed to Poirier-Ginter et

al. [54], no multi-view consistency optimization was performed: we simply used vanilla 3DGS directly on the images relit by SyncLight.

I Additional visual results

Fig. 16 shows further qualitative results of competing methods and SyncLight across the Infinigen and BlenderKit splits of our dataset (Real split shown in the main paper), demonstrating the strong performance of our method. In contrast, Flux.2-dev struggles with cross-view consistency, producing visually plausible but geometrically inconsistent results — note the differing lamp colors across views in BlenderKit row 2. The LightLab*+LumiNet baseline, while maintaining better consistency by explicitly relighting each view, fails to propagate lighting effects when sources are not directly visible (BlenderKit row 2, second view).

Fig. 18–fig. 19 present additional results from our method on the RealEstate10K dataset, illustrating various scenarios in which different lights are switched on or off under different colors and intensity conditions.

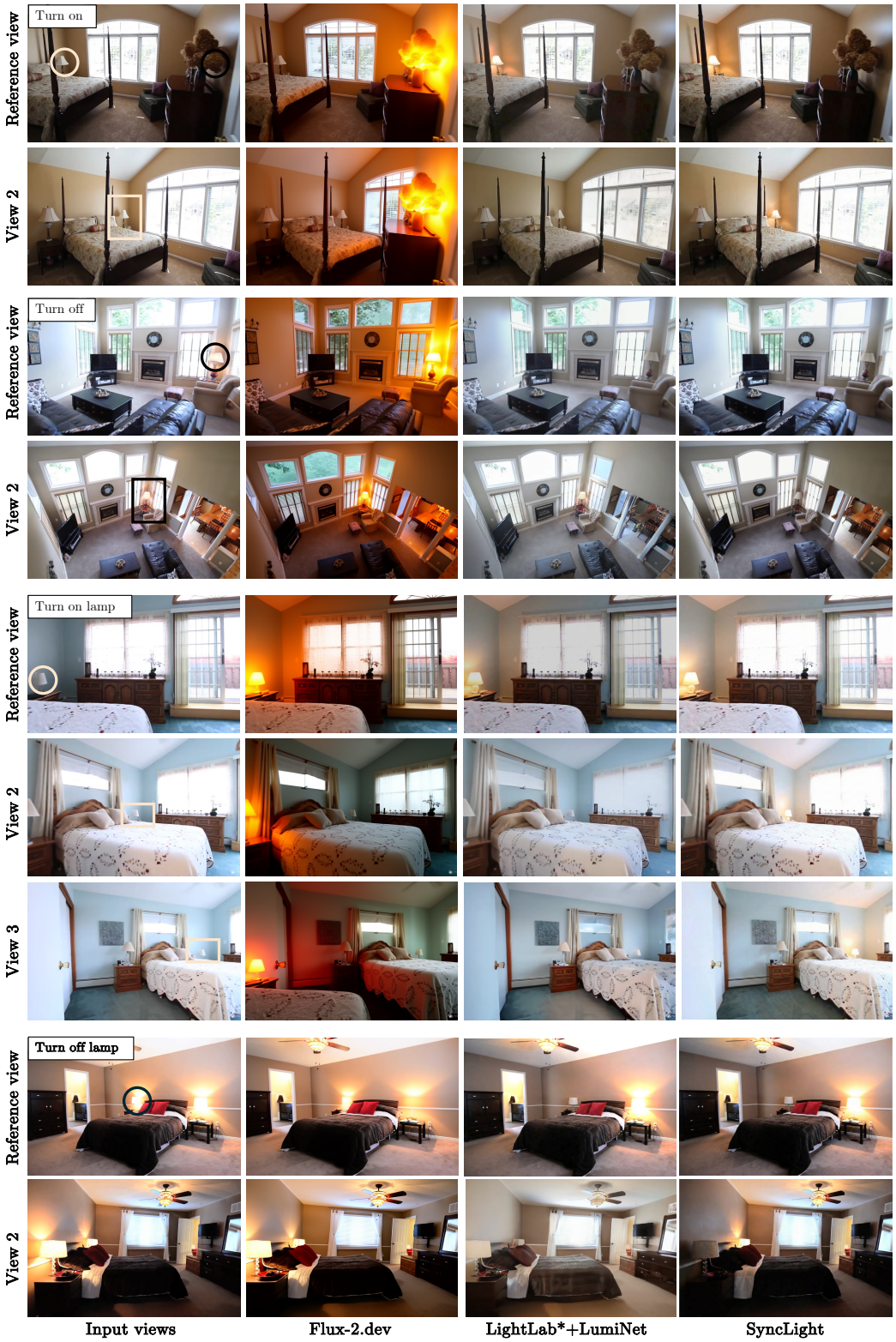


Figure 18: Additional results on the RealEstate10K dataset.

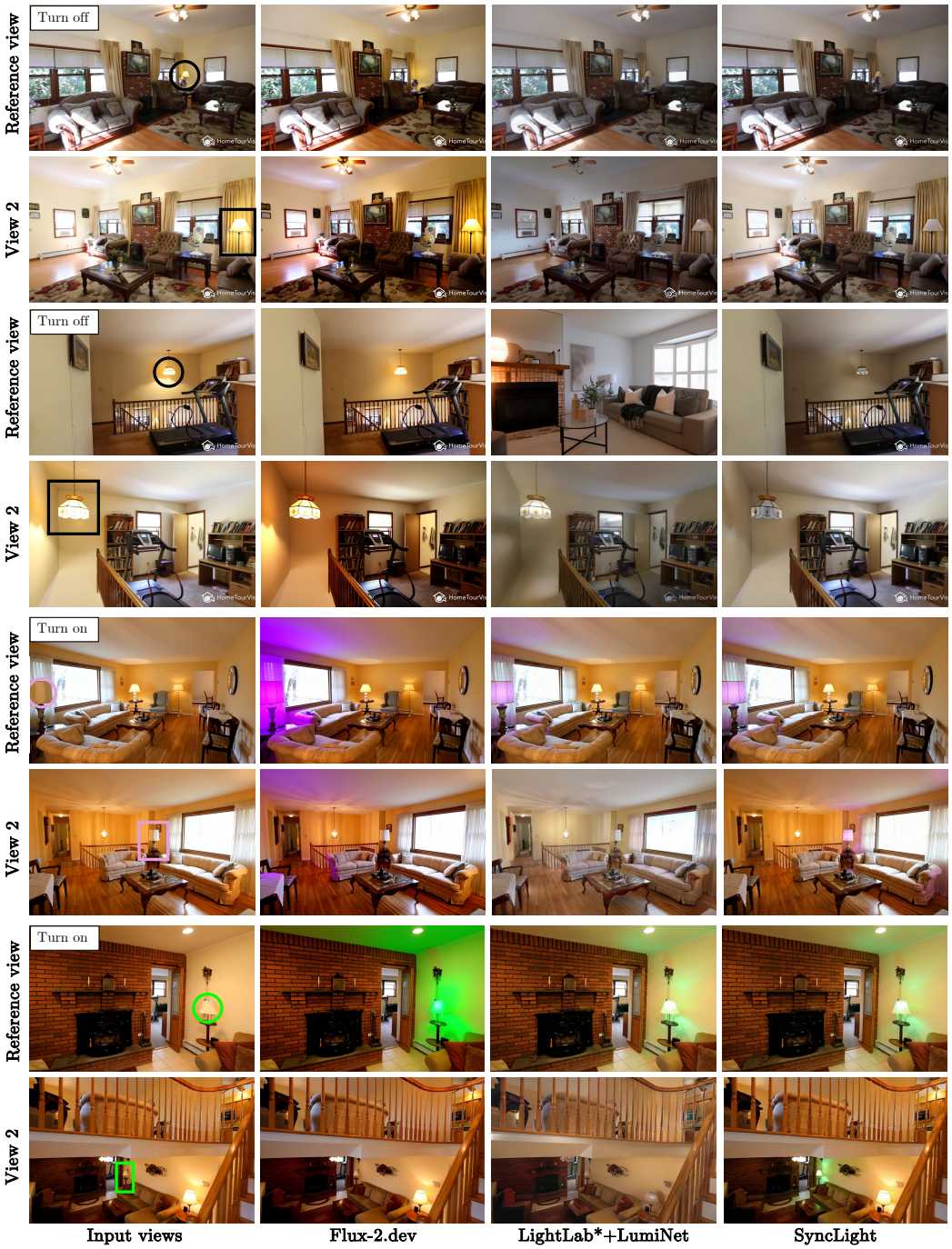


Figure 19: Additional results on the RealEstate10K dataset

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are backed up in the following sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed on the last section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: There are not theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All details for replicating the model are explained in Section 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We are committed to release the code, models and data upon acceptance, but it has not been provided in the submission itself.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All the experimental details are provided in Section 5 and supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: We report error bars for our user study and the standard deviation of inference-time measurements, computed across multiple images.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5 details the GPUs used and the inference time for each method.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our papers conforms with the NeurIPS Code for Ethics.

   Guidelines: We have reviewed the NeurIPS Code of Ethics and confirm that our research conforms with it. The work uses publicly available datasets and self-captured indoor scenes containing no personally identifiable information.

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes, last section discusses broader impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our model performs lighting-only edits on input images, modifying illumination conditions without altering scene content. The model cannot be used to alter any non-lighting attributes of a scene, which substantially limits its potential for misuse compared to general-purpose image generation models. We therefore consider the risk of misuse to be low.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Any asset used in our new dataset is properly credited in Section 4.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new dataset is explained in Section 4.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We provide the information and the screenshot in the supplementary.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: They were not used.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.