

RenderFlow: Single-Step Neural Rendering via Flow Matching

- Supplementary Material -

Shenghao Zhang^{1,2} Runtao Liu¹ Christopher Schroers¹ Yang Zhang¹

¹Disney Research|Studios ²ETH Zürich

1. Implementation Details

1.1. Forward Rendering

Our model is fine-tuned from the pre-trained Wan2.1 (1.3B) video diffusion model [8]. Given that our task is conditioned on G-buffers rather than text, we remove the original cross-attention layers from the transformer blocks. We adapt the input embedder from VACE [2], modifying it to process albedo interpolants instead of noisy inputs. The VACE context embedder is repurposed to process both the G-buffer attribute tokens and the progressive predictions from previous video chunks by expanding its input channel dimension. For environment map conditioning, we employ a separate input embedder, initialized with the same weights as the primary albedo embedder. Keyframe features are processed frame-wise using dedicated 2D convolutional layers to patchify the keyframe latents into spatially-aligned tokens.

The training process consists of two main stages. To manage the significant memory consumption of the VAE decoder during pixel-space loss computation, we train our model on short sequences of 5 frames. The initial training stage consists of 20,000 steps at a 256×256 resolution with a batch size of 32 and a constant learning rate of 5×10^{-5} . We then continue training for another 20,000 steps at the full 512×512 resolution, using a learning rate of 3×10^{-5} with a 1,000-step warm-up period. For the final keyframe guidance stage, the model is trained for an additional 20,000 iterations at 512×512 with a learning rate of 1×10^{-4} . All training and experiments utilize the AdamW [4] optimizer. Each training stage takes approximately two days to complete on four NVIDIA A100 GPUs.

1.2. Inverse Rendering

Architecture Details. We adapt the pretrained *RenderFlow* framework for inverse rendering by freezing the entire forward backbone—including the VAE encoder/decoder (\mathcal{E}, \mathcal{D}) and the DiT transformer blocks—and introducing a set of parameter-efficient adapter modules. First, we replace the original G-buffer input projection with a trainable in-

verse embedder. This module maps the latent representation of the input RGB image, $z_{\text{rgb}} = \mathcal{E}(I_{\text{rgb}})$, into the token space required by the transformer. Second, to adapt the frozen self-attention mechanisms for decomposition tasks, we inject Low-Rank Adaptation (LoRA) modules into the query, key, and value projections of each block. Third, to control the target output modality (e.g., Albedo vs. Normal), we incorporate a prompt-based cross-attention layer after each self-attention block. Following RGB-X [9], we use text embeddings of the target attribute as keys and values, allowing the model to switch tasks dynamically. Finally, the transformer outputs are processed by lightweight, task-specific intrinsic heads (MLPs) that project the features into the appropriate latent space before decoding to further distinguish between modalities.

Training Objectives. We utilize the same bridge matching framework as the forward pass but apply it to the latent trajectory from the RGB input to the target intrinsic. We supervise the training using modality-specific reconstruction losses \mathcal{L}_{rec} calculated in the pixel space:

1. **Albedo:** We utilize a combination of \mathcal{L}_1 distance and a perceptual loss (LPIPS) to preserve both color accuracy and high-frequency details:

$$\mathcal{L}_{\text{albedo}} = \|\hat{I} - I_{\text{gt}}\|_1 + \lambda \mathcal{L}_{\text{LPIPS}}(\hat{I}, I_{\text{gt}}) \quad (1)$$

2. **Normal:** We employ a cosine similarity loss to enforce angular consistency between the predicted and ground truth normal vectors:

$$\mathcal{L}_{\text{normal}} = 1 - \frac{1}{N} \sum \frac{\langle \hat{I}, I_{\text{gt}} \rangle}{\|\hat{I}\|_2 \cdot \|I_{\text{gt}}\|_2} \quad (2)$$

3. **Depth:** To handle scale ambiguity, we use the scale-and-shift invariant (SSI) loss [6]:

$$\mathcal{L}_{\text{depth}} = \frac{1}{N} \sum \Delta_i^2 - \frac{1}{2N^2} \left(\sum \Delta_i \right)^2 \quad (3)$$

where Δ_i is the difference in log-depth at pixel i .

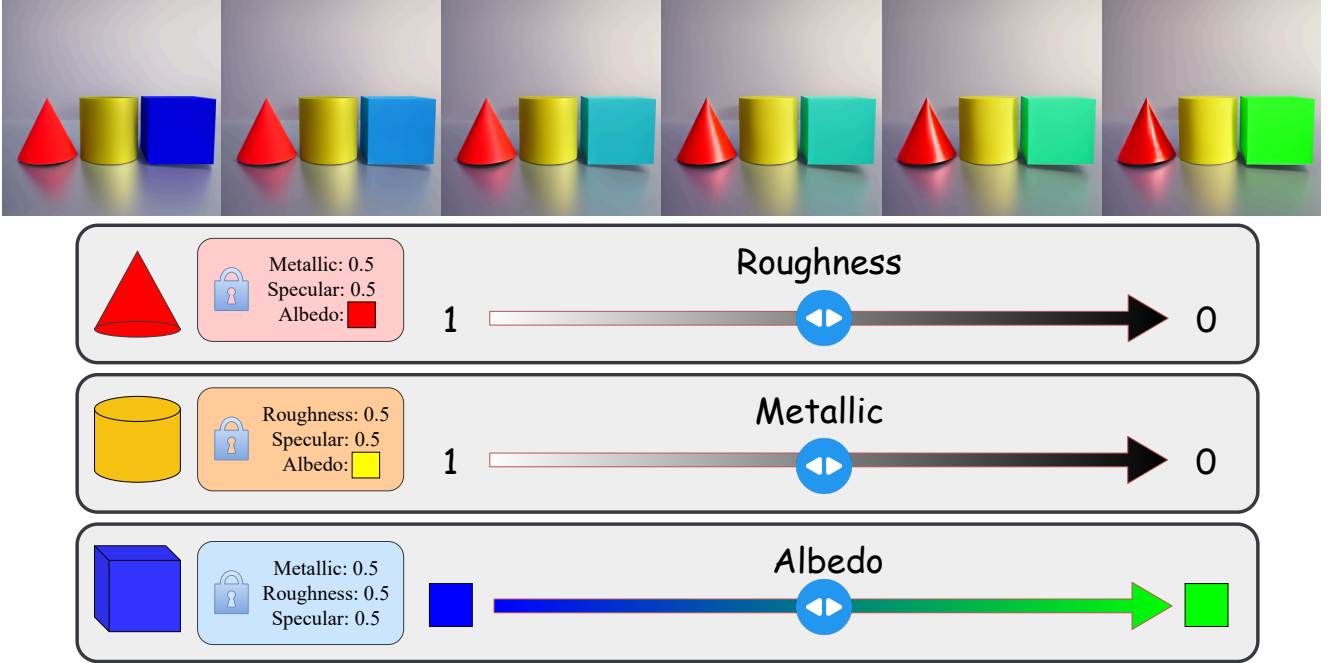


Figure 1. **Demonstration of controlled material editing.** Our model is capable of synthesizing smooth transitions for individual material properties while all other scene parameters are held constant. From left to right: the cone’s roughness is interpolated from 1.0 to 0.0; the cylinder’s metallic property from 0.0 to 1.0; and the cube’s diffuse albedo from blue to green.

4. **Material:** For roughness and metallic maps, we apply a standard \mathcal{L}_1 loss:

$$\mathcal{L}_{\text{material}} = \|\hat{I} - I_{\text{gt}}\|_1 \quad (4)$$

We train the adapter modules on the same dataset used for the forward model. The backbone weights remain frozen throughout the process. We use the AdamW optimizer with a learning rate of 1×10^{-4} and a total batch size of 32. The model is trained for 20,000 iterations on 4 NVIDIA A100 GPUs. During training, we randomly sample a target modality and its corresponding text prompt for each batch to ensure the model learns to disentangle all intrinsics simultaneously.

1.3. Input Buffer Selection

We evaluate the choice of primary input buffer by comparing Albedo alone against a combined Albedo×Irradiance input. Both variants are trained at low resolution for one day as a lightweight ablation. As shown in Table 1, incorporating irradiance as a multiplicative prior improves PSNR by 1.9 dB, confirming that pre-multiplied lighting information provides a useful signal. However, to maintain a fair comparison with DiffusionRenderer [3], which uses albedo as its primary input, we adopt the same protocol in our main experiments.

Input	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
Albedo	23.7	0.75	0.13
Albedo×Irradiance	25.6	0.76	0.13

Table 1. **Buffer combination analysis.** Trained at low resolution for one day as a lightweight ablation. Combining albedo with irradiance as a multiplicative prior improves reconstruction quality.

2. Evaluation of Keyframe Guidance

We further evaluate the effectiveness of our keyframe guidance by analyzing its performance under varying levels of keyframe sparsity. While the model was trained with a fixed keyframe gap of 16 frames, the adapter’s design allows for flexible injection of arbitrary keyframes during inference. We conduct experiments with keyframe gaps of 13, 17, 25, and 49 frames, and include a baseline without keyframe guidance. Table 2 summarizes the quantitative results. The experiment shows that, consistent with the intuition, increasing the keyframe gap leads to a degradation in reconstruction quality. This confirms that more frequent keyframes provide more effective guidance. Notably, even with a large keyframe gap of 49 frames, the model with keyframe guidance still significantly outperforms the unguided baseline, demonstrating the robustness of the guidance mechanism. The consistency achieved by the progres-

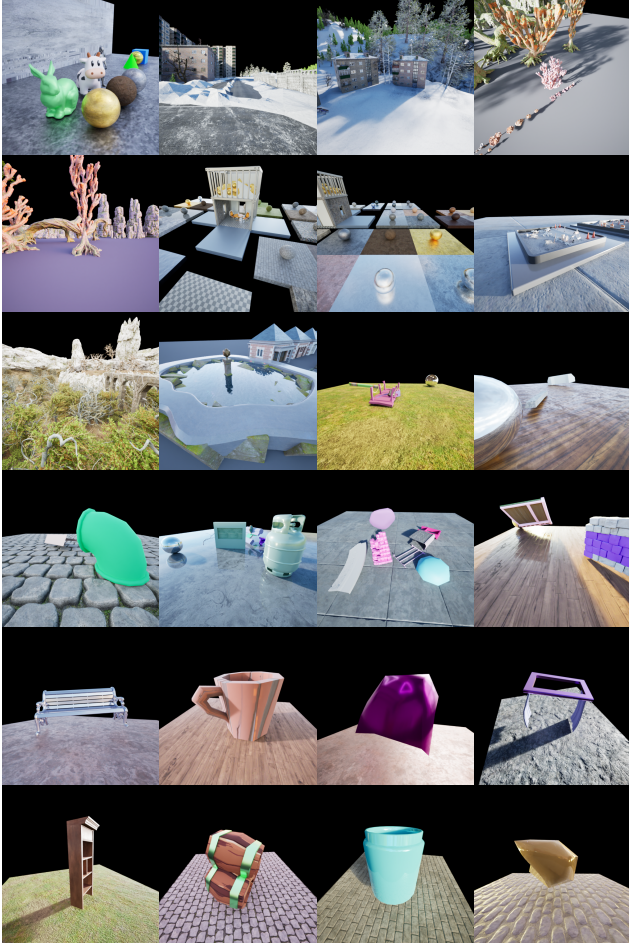


Figure 2. A preview of our synthesized dataset, showcasing a variety of scenes, objects, and lighting conditions.

sive inference is shown in the **Supplementary Videos**.

Method	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
w/o keyframes	24.022	0.899	0.112
13 Gap	29.716 (29.427)	0.920 (0.918)	0.089 (0.091)
17 Gap	29.071(28.812)	0.917(0.916)	0.092(0.094)
25 Gap	26.574(26.507)	0.913(0.911)	0.098(0.099)
49 Gap	25.917(25.737)	0.909(0.909)	0.102(0.104)

Table 2. **Quantitative evaluation of the impact of keyframe gaps.** Reconstruction quality degrades as the distance between keyframes increases. Values in parentheses report metrics calculated exclusively on non-keyframe frames to evaluate guidance performance.

3. Demonstration of Material Editing

In this section, we evaluate our model’s ability to render materials in a controlled and disentangled manner. We conducted a synthetic experiment for plausible material editing. We generated image sequences where specific material properties were dynamically varied for individual objects, while all other scene parameters, such as camera path and environment lighting, remained fixed.

The scene, shown in Figure 1, contains three primitive objects: a red cone, a yellow cylinder, and a blue cube. Throughout the sequence, we linearly interpolate a single material parameter for each object while holding its other properties constant:

- **Cone (Roughness):** The cone’s diffuse albedo is fixed to red, with metallic and specular values all set to 0.5. Its roughness parameter is linearly interpolated from 1.0 (fully diffuse) to 0.0 (perfectly smooth).
- **Cylinder (Metallic):** The cylinder’s roughness and specular
- **Cone (Roughness):** The cone’s diffuse albedo is fixed to red, values are fixed at 0.5 and 0.5. Its metallic parameter is linearly interpolated from 0.0 (dielectric) to 1.0 (fully metallic).
- **Cube (Albedo):** The cube’s roughness and metallic values are fixed at 0.5 and 0.5. Its diffuse albedo is linearly interpolated in RGB space from blue (0,0,1) to green (0,1,0).

The results illustrate that our model can faithfully render the smooth transitions between material properties, synthesizing the changing appearance from diffuse to specular reflections on the cone, dielectric to conductive properties on the cylinder, and the color shift on the cube. This highlights the model’s potential for downstream applications, such as interactive material editing.

4. Analysis on inference steps

Our model is trained in a 4-step bridge matching framework, making it possible to either choose to perform single-step or multi-step (2 or 4 steps) inference. In our experiments, we empirically found that single-step inference consistently outperforms multi-step inference across all quantitative metrics. We attribute this to error propagation: our model is trained to predict the final rendered latent \hat{z}_1 directly from the initial albedo latent z_0 . Due to the inaccuracy of each network evaluation, intermediate errors will be accumulated, resulting in greater offsets from the ground truth.

Qualitatively, as shown in Figure 3(a), we observe that multi-step inference can produce visually sharper high-frequency details, such as contact shadows. We hypothesize that this apparent sharpness comes at the cost of geometric and photometric accuracy. The iterative process may

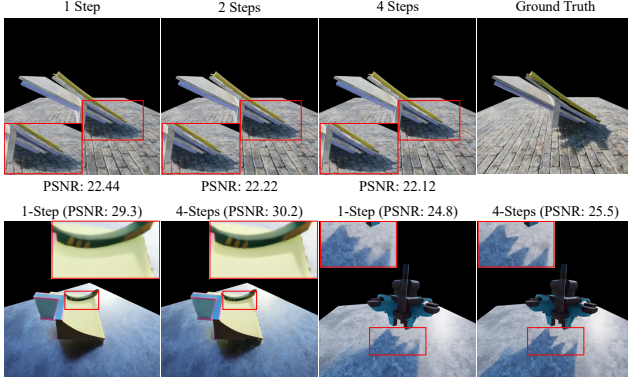


Figure 3. **Analysis on inference steps.** (a) Top: Multi-step inference can generate sharper shadows but leads to lower metrics due to misalignments with ground truth. (b) Bottom: In favorable cases where generated content aligns with ground truth, 4-step inference yields sharper and more detailed results.

amplify certain features, making them visually prominent but causing them to deviate from the ground truth. Therefore, while multi-step inference can create plausibly sharp details, the single-step approach achieves a better balance of visual quality and quantitative accuracy. Nevertheless, in cases where the generated content aligns well with the ground truth, 4-step inference can produce better results both visually and quantitatively. Figure 3(b) illustrates such a case, where 4-step inference yields sharper and more detailed outputs compared to single-step inference.

5. Dataset Curation

Our dataset was synthesized using the *Movie Render Queue* in Unreal Engine 5. The dataset is composed of two main sources: pre-existing, artist-crafted environments and procedurally generated synthetic scenes. The artist-crafted environments were sourced from the Unreal Engine Marketplace [7] and include professionally built scenes with high-quality assets. To adapt these scenes for our task, we removed all explicit light sources, relying solely on image-based lighting from HDR environment maps. We also manually modified materials to eliminate transparent and translucent surfaces, ensuring that the G-buffers provide a complete and unambiguous representation of the scene geometry.

Rendering these complex, pre-existing environments is time-intensive, often taking over one hour for a 100-frame sequence. To significantly increase data diversity and volume, we developed a procedural pipeline to generate synthetic scenes. This pipeline randomly places objects from a curated asset collection onto a ground plane and assigns them dynamic materials with randomized properties. This automated approach reduced rendering time to approxi-

mately 15 minutes per 100-frame sequence. For the artist-crafted environments, we defined fixed camera trajectories, whereas for the synthetic scenes, we primarily utilized orbital camera paths. To further diversify lighting conditions, we augmented the environment maps from the pre-existing scenes with additional HDRIs from Poly Haven [5].

In total, our asset library contains over 4,000 unique meshes and more than 30 HDR environment maps. The final dataset consists of approximately 30,000 frames from artist-crafted environments and 100,000 frames from synthetic scenes. All frames were rendered at a resolution of 512x512 pixels with 256 samples per pixel (SPP) and subsequently denoised using Intel Open Image Denoise [1]. While our dataset is smaller than that used by Diffusion-Renderer [3], which contains 150,000 videos, our model’s efficiency and training strategy allow it to achieve strong performance. A preview of our dataset is provided in Figure 2.

6. Limitations and Future Work.

While RenderFlow demonstrates significant advancements, several avenues for improvement remain. First, the generalization of our model is limited by the diversity of our current dataset. Expanding the training data to include a broader range of lighting phenomena and geometric complexity would enhance the model’s robustness. Second, the VAE’s latent space can cause information loss, potentially limiting the reconstruction of details in highly complex geometries as shown in Figure 4. Furthermore, the causal convolution within the Wan VAE encoder performs temporal compression that affects frame consistency. Specifically, later frames tend to exhibit progressive blurring compared to the sharp initial frame due to causal dependencies in the latent space as shown in Figure 5. Finally, the VAE encoder and decoder still constitute a significant computational bottleneck. Future work could explore more efficient encoding strategies for G-buffers to enhance quality and speed. We believe that with larger and more diverse datasets, more efficient model architectures, our proposed paradigm has the potential to achieve greater performance in neural rendering and video synthesis.

7. Additional Qualitative Results

We provide additional qualitative comparisons in Figures 6 and 7, including error maps to highlight differences between the ground truth and our predictions. The results show that our model achieves performance comparable to Diffusion-Renderer in synthesizing complex effects like shadows and detailed reflections, while better preserving the underlying geometry of the scene. Resolving misalignments in shadows and reflections is challenging, as the information provided by G-buffers alone can be ambiguous. However, we

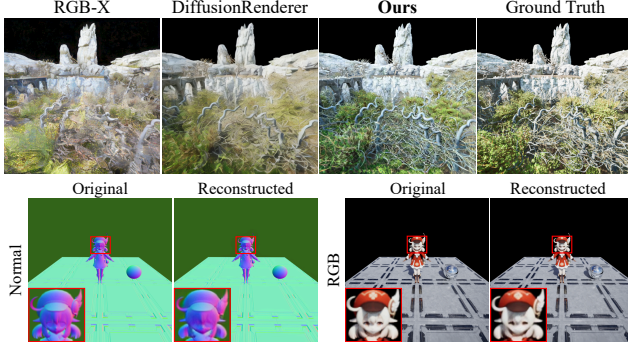


Figure 4. **Failed cases and analysis.** Top: Rendered images for scenes with highly complex geometries. Bottom: The VAE compression leads to loss of fine-grained details

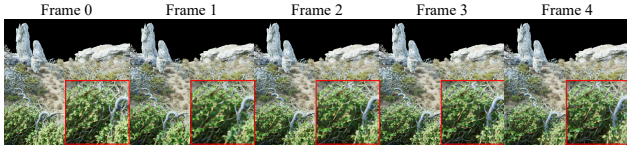


Figure 5. **Analysis on video latent downsampling.** The causal design of the VAE encoder leads to the phenomenon that the initial frame retains sharpness while subsequent frames exhibit progressive blurring due to temporal downsampling dependencies.

demonstrate that with keyframe guidance, our model can leverage information from the reference view to generate these high-frequency details with greater physical plausibility and accuracy.

References

- [1] Attila T. Áfra. Intel® Open Image Denoise, 2025. <https://www.openimagedenoise.org>. 4
- [2] Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. Vace: All-in-one video creation and editing. *arXiv preprint arXiv:2503.07598*, 2025. 1
- [3] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 4
- [4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1
- [5] Poly Haven. Poly Haven: The public 3d asset library. <https://polyhaven.com/>, 2025. Accessed: August 3, 2025. 4
- [6] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. 1
- [7] Unreal Engine. Unreal engine marketplace. <https://www.fab.com/channels/unreal-engine>, 2025. Accessed: August 3, 2025. 4

- [8] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Fei Wu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingtong Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wenten Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 1
- [9] Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. Rgbx: Image decomposition and synthesis using material- and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*, New York, NY, USA, 2024. Association for Computing Machinery. 1

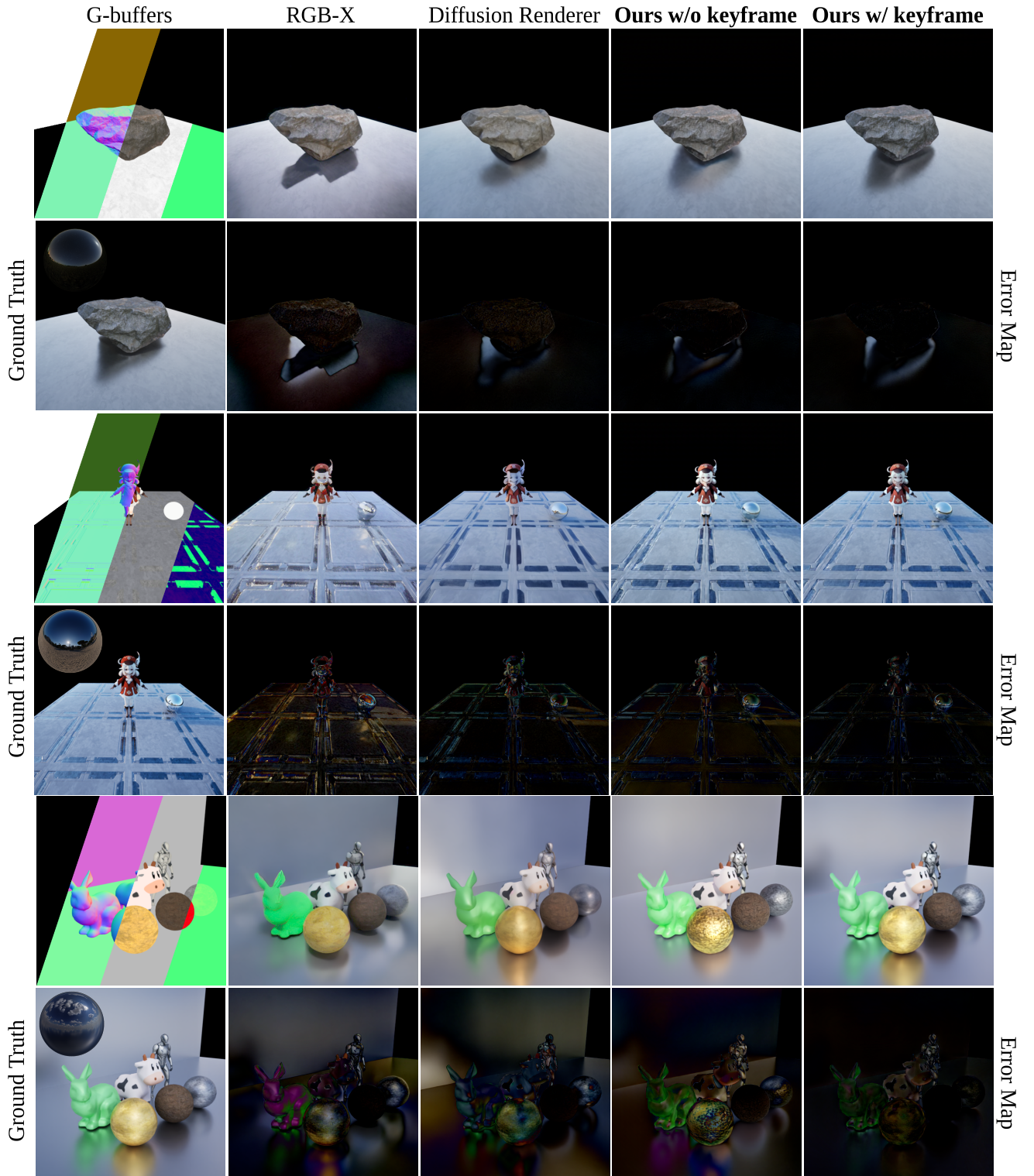


Figure 6. **Additional qualitative forward rendering results.** We provide more qualitative comparisons with error maps to highlight the differences between our predictions and the ground truth. For keyframe guidance, we sample keyframes with a 25-frame gap. Our method clearly outperforms the baseline methods. In addition, with the keyframe guidance our model achieves the best results among all models.

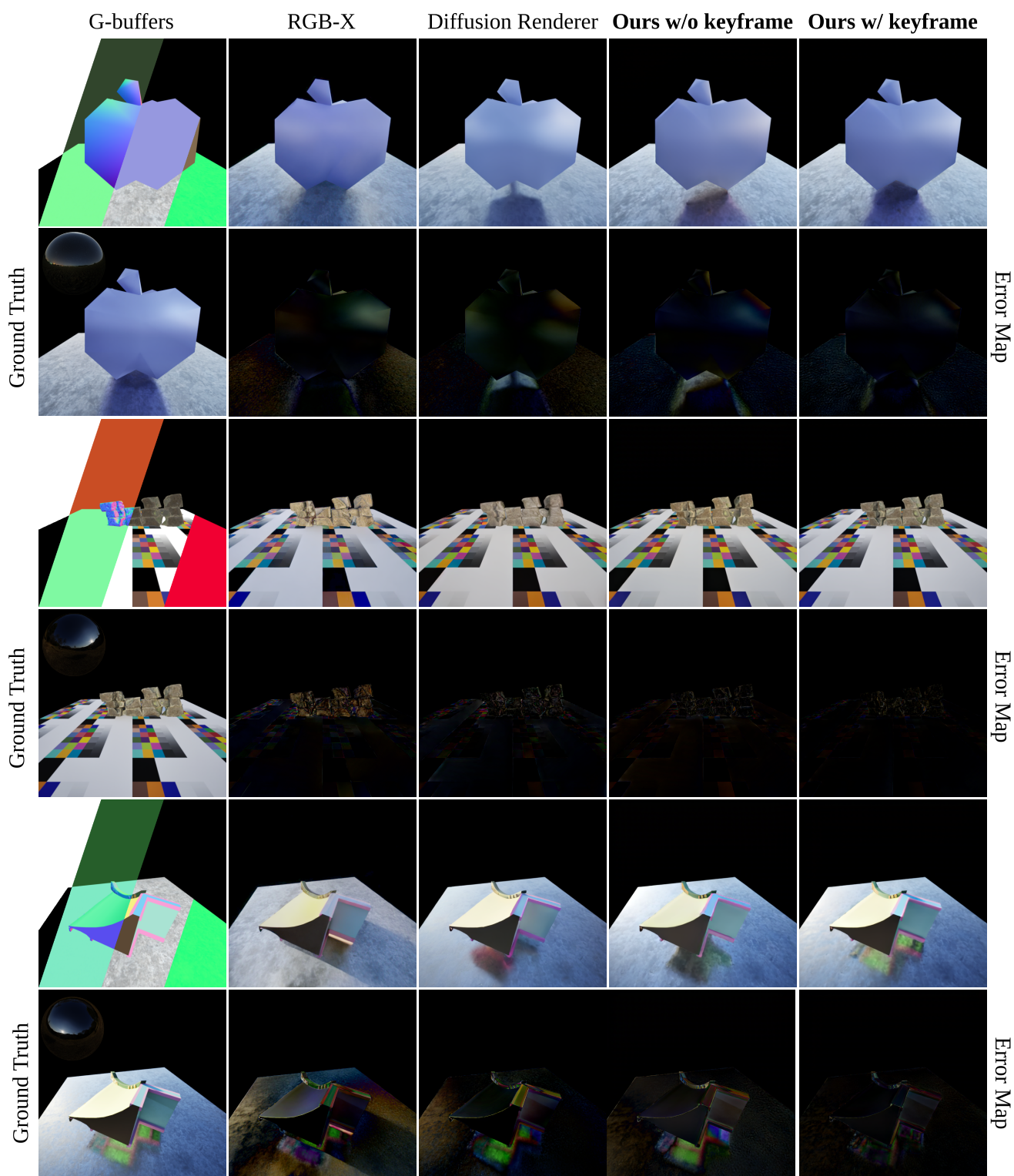


Figure 7. Additional qualitative forward rendering results (continued).

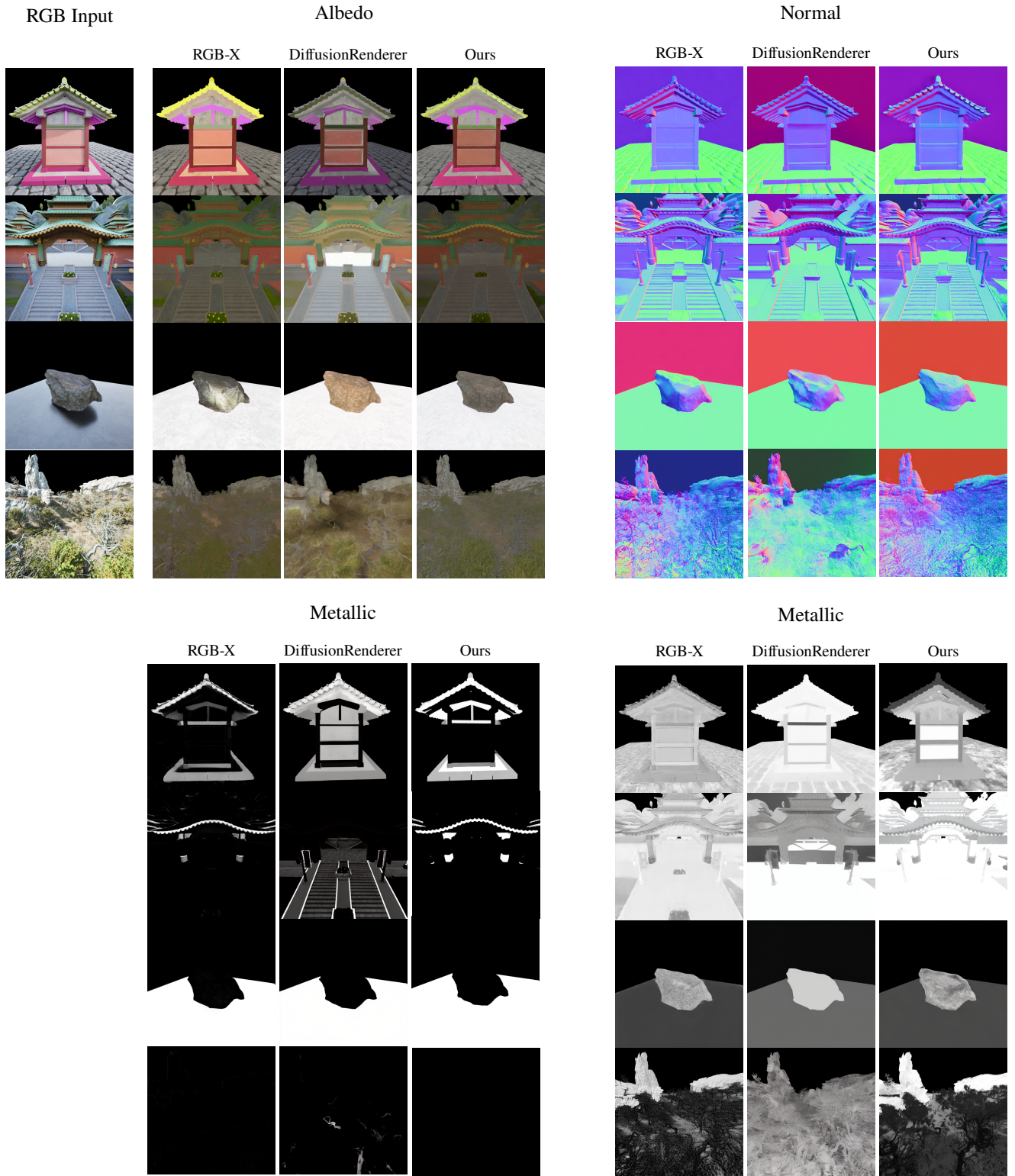


Figure 8. Additional qualitative inverse rendering results.