IntrinsicControlNet: Cross-distribution Image Generation with Real and Unreal

Supplementary Material

1. More Evaluation Results

Evaluate on synthetic datasets. For synthetic data, we compared our method to several popular graphics engines, including UE [7], Falcor [2], and Blender [3], to evaluate image synthesis results. For our method, we employ the GT intrinsic images as input conditions to generate the target image, while for rendering engines, we utilize complete 3D scenes to render the target images. As shown in Figure 1, various methods produced highly realistic photorealistic images. However, compared to the images generated by various engines, our method produces images that more closely align with the realism of human cognition. To better quantify the realism of the generated images, we additionally computed the CLIP (Contrastive Language-Image Pre-training) score [19, 20] and conducted a user study for generated results(refer supplementary for details). We present the quantitative analysis results in Table 2, showing that our method outperforms others in CLIP scores, LPIPS scores, FID scores, and user studies.

Evaluate on real-world datasets. We evaluated IntrinsicControlNet with untrained real-world data from the ADE dataset, including indoor and outdoor data. Specifically, we first predicted intrinsic images from real-world images and used them as inputs to generate new images, then compared the generated images with the original ones to evaluate their similarity. As shown in Figure 1, even though the input intrinsic images are biased due to prediction, IntrinsicControlNet can still generate photorealistic images that closely resemble the originals.

2. Training Detail

Our model is trained using 4 NVIDIA A6000 GPUs with a batch size of 48 for 280k iterations based on Stable Diffusion v2.1 pre-trained model [22]. For the inference process, we adopt the DDIM sampler [23] with 50 sampling steps by using a single NVIDIA A6000 GPU. We use the AdamW optimizer with a fixed learning rate of 1×10^{-5} [15] and weight decay of 0.01. During training, we center-croped the image with 512 × 512 resolution. Empirically, $\lambda_{\text{realistic}} = 0.2$, $\lambda_{\text{gen}} = 1$, $\lambda_{\text{tg}} = 1$, and $\lambda_{\text{dis}} = 0.5$

3. Datasets Preparation

Dataset composition. To ensure our model is versatile and capable of handling diverse scene generation tasks, we train it using a comprehensive mixed dataset. Table 1 provides details on the composition, size, and sources of the various components within this dataset. This mixed dataset includes a wide range of data types, such as real-world, synthetic, indoor, outdoor, single-object datasets, as well as datasets for embodied AI and autonomous driving. The diverse intrinsic image sources contribute to the robustness of our model, while the inclusion of synthetic datasets allows for better control over detailed structures and colors. Meanwhile, real-world datasets ensure that the results produced by our model possess a high degree of realism. This enriched mixed dataset enhances our model's accuracy and robustness in fitting both geometry and material properties.

4. Metrics and User Study

4.1. CLIP Score

The CLIP score [8] is calculated by determining the cosine similarity between image and text embeddings, making it particularly useful for tasks that require cross-modal understanding. In our task, we employ the BLIP model [13] to extract a text prompt from various images, including scene images synthesized by the graphics engine and real-world photo images. We then enhance this prompt with the keywords 'photo-realism' to generate a comprehensive text description. We compute the CLIP score between this generated text description and the images produced by each method.

4.2. LPIPS Score

LPIPS (Learned Perceptual Image Patch Similarity) [31] is a metric designed to evaluate the perceptual similarity between images. Unlike traditional pixel-level comparison methods, LPIPS assesses differences by analyzing deep features extracted from the images. It utilizes a pre-trained convolutional neural network, such as AlexNet, to obtain these features and then computes the distance between them.

4.3. DINOv2 Similarity Distance

DINOv2 similarity distance. DINOv2 similarity distance [17] is a metric used to measure the similarity between images, based on features extracted by the DINOv2 model. We use this metric to evaluate the similarity between the results generated by different methods and various architectural designs of our model with the reference images.

DINOv2 self-similarity distance. We also employed the DINOv2 self-similarity distance [17] to assess the consistency between different parts of the image resulting from object insertion. This metric helps evaluate whether the inserted object integrates cohesively with the overall image, specifically examining if it aligns with the lighting and

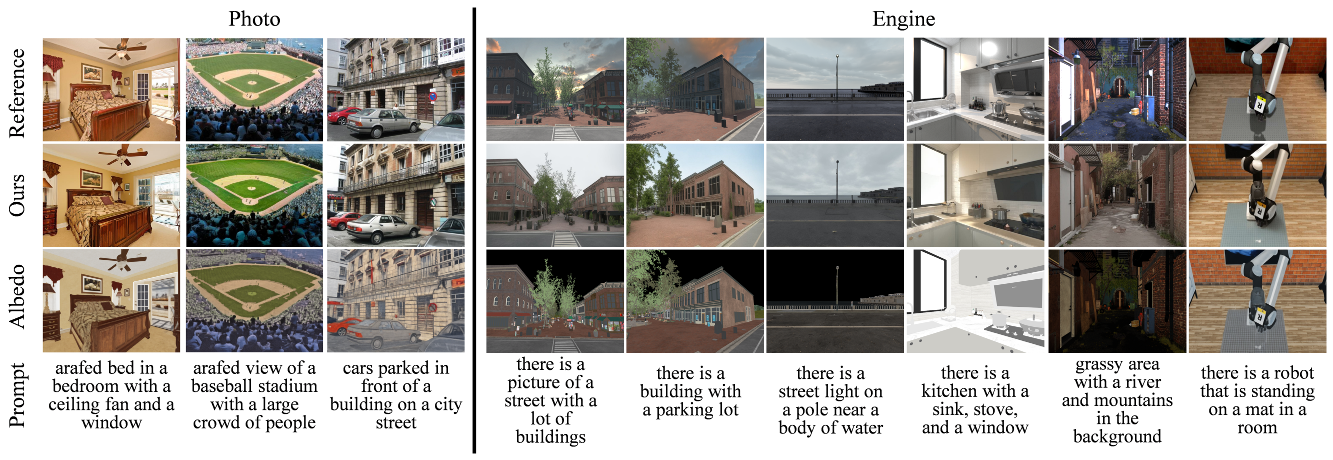


Photo
Engine
Reference
Ours
Albedo
Prompt
arafed bed in a bedroom with a ceiling fan and a window
arafed view of a baseball stadium with a large crowd of people
cars parked in front of a building on a city street
there is a picture of a street with a lot of buildings
there is a building with a parking lot
there is a street light on a pole near a body of water
there is a kitchen with a sink, stove, and a window
grassy area with a river and mountains in the background
there is a robot that is standing on a mat in a room

Figure 1. Comparing results of our framework on real-world and synthetic datasets.

Datasets	Real	Synthetic	In.	Out.	Size	A.	R.	M.	N.	D.	S.	L.
ADE [33]	✓		✓	✓	8k	✓	✓	✓	✓	✓	✓	StyleLight [25]
PanoContext [32]	✓		✓		0.7k	✓	✓	✓	✓	✓		✓
Stanford 2D-3D-S [1]	✓		✓		0.3k	✓	✓	✓	✓	✓		✓
Bridge Data v2 [24]	✓	✓	✓		2k	✓	✓	✓	✓	✓		
OpenDV-YouTube [26]	✓			✓	8k	✓	✓	✓	✓	✓		
InteriorVerse [34]		✓	✓		48k	✓	✓	✓	✓	✓		✓
GTA-V [21]		✓		✓	3k	✓	✓	✓	✓	✓		
Poly Haven [11]	✓			✓	1k	✓	✓	✓	✓	✓		✓
GObjaverse [18]		✓	single object		21k	✓	✓		✓			✓
Our Captures	✓		✓	✓	4k	✓	✓	✓	✓	✓		✓

Table 1. Datasets overview. 'A.', 'R.', 'M.', 'N.', 'D.', 'S.', 'L.' represent albedo, roughness, metallicity, normal, depth, semantic segmentation, and lighting. "In." and "Out." denote whether the scene is indoor or outdoor. "Size" indicates the number of images contained in the dataset. All intrinsic images for real data are predicted, while all intrinsic images for synthetic data are obtained as ground truth from the rendering engine. Moreover, We refine the method in [12] on outdoor datasets, enabling it to acquire intrinsic decomposition capabilities for outdoor scenes.

shadow effects of the background.

4.4. L2 Distance

L2 distance is a metric for measuring the straight-line distance between two points, commonly used in the field of image similarity to quantify and compare the differences between image feature vectors to assess the similarity of images.

4.5. FID Score

FID (Fréchet Inception Distance) [9] is a metric that quantifies the similarity between real and generated images by comparing their feature distributions, used to assess the quality of generative models in image synthesis tasks. We evaluate image realism using FID and IS metrics on 3k real images from the ADE validation set and IIW dataset. Each method generates 3k images from randomly selected untrained data in a mixed dataset (see Table 1). Our method significantly outperforms others in FID metric.

4.6. User Study

We conducted a user study to evaluate the realism of images produced by various methods, including graphics engines, Multi-ControlNet, RGB↔X [28], and Ctrl-X [14]. For each graphics engine, we collected 4 scenes and rendered 8 images from different viewpoints for each scene. Figure 1 and Table 2 present the generated results and quantitative metrics for 5 of these scene images, naming 'Buildings', 'Corner', 'Street', 'Kitchen', and 'Alley' respectively. We also expand our user study on 15 real-world and 15 synthetic embodied AI images, with their intrinsic images obtained through prediction. Therefore, Our user study involved a total of 218 image pairs. For each pair, participants were asked to rate which image appeared most realistic. We gathered 28 valid questionnaires, and the results are presented

Table 2. Quantitative comparison of different methods on synthetic and read-world datasets."US" is the abbreviation for "User study". In *Engine* part in 1, the first three reference images are rendered using UE, the fourth with Falcor, and the last two with Blender.

Scene	Buildings			Corner			Street			Kitchen			Alley			Real			All
Scores	CLIP ↑	LP. ↓	US ↑	CLIP ↑	LP. ↓	US ↑	CLIP ↑	LP. ↓	US ↑	CLIP ↑	LP. ↓	US ↑	CLIP ↑	LP. ↓	US ↑	CLIP ↑	LP. ↓	US ↑	FID ↓
Ours	**0.267**	**0.381**	**92.9**	**0.299**	**0.437**	**78.6**	**0.261**	**0.258**	**53.6**	0.286	0.387	**39.3**	**0.307**	**0.461**	**89.3**	**0.293**	0.363	**67.9**	**26.40**
Multi.	0.251	0.564	0.00	0.239	0.532	0.00	0.253	0.335	14.3	0.269	0.461	3.57	0.247	0.479	0.00	0.239	0.368	7.14	48.69
RGB↔X [28]	0.233	0.464	3.57	0.270	0.485	21.4	0.240	0.459	7.14	0.281	**0.378**	21.4	0.282	0.471	0.00	0.280	**0.357**	25.0	42.90
Ctrl-X [14]	0.252	0.678	0.00	0.250	0.665	0.00	0.226	0.727	0.00	0.270	0.587	0.00	0.265	0.670	0.00	0.247	0.678	0.00	62.63
UE [7]	0.252	-	3.57	0.230	-	0.00	0.251	-	25.0	-	-	-	-	-	-	-	-	-	-
Falcor [2]	-	-	-	-	-	-	-	-	-	**0.292**	-	35.7	-	-	-	-	-	-	-
Blender [3]	-	-	-	-	-	-	-	-	-	-	-	-	0.245	-	10.7	-	-	-	-

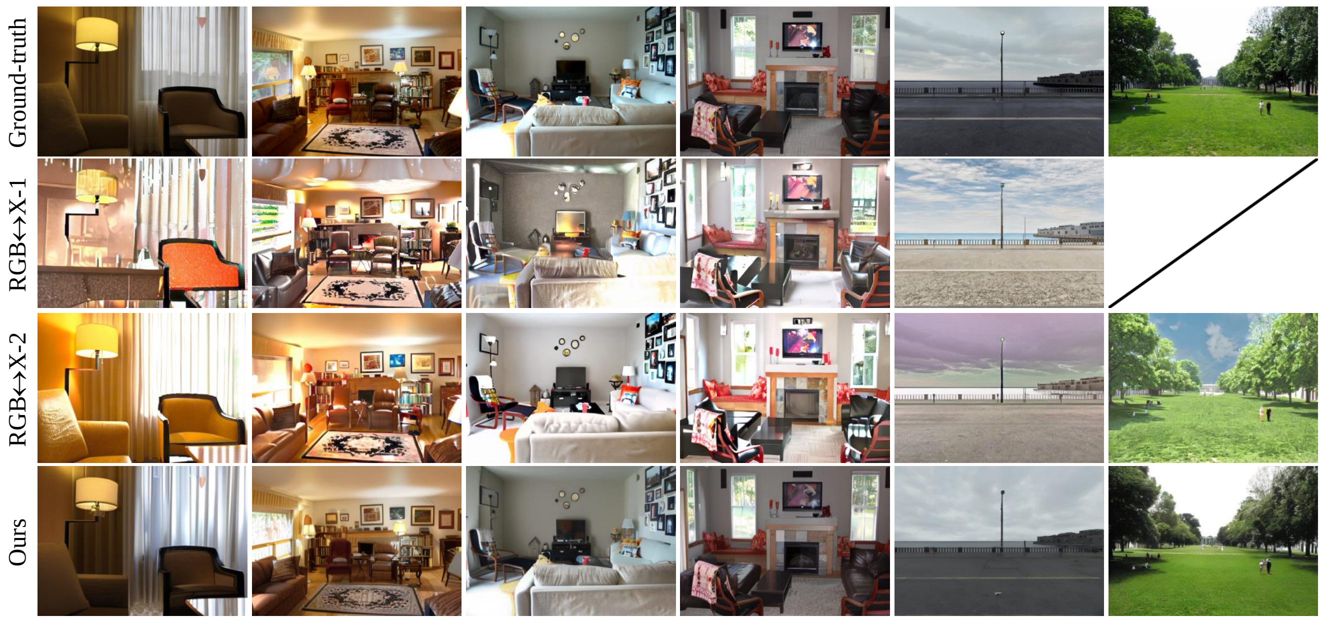


Figure 2. Comparsion between IntrinsicControlNet and RGB↔X [28]. The second row shows the results using the intrinsic images provided by the dataset. The third row shows the results of intrinsic images decomposed by the RGB→X process of RGB↔X and then re-rendered through the X→RGB process. The final row displays the results from our model using the dataset-provided intrinsic images as conditional input. The last column represents the real data, so both we and RGB↔X use our respective material decomposition methods to obtain intrinsic images, which were then re-rendered to produce the results.

in Table 2, which shows the percentage of participants who found images generated by each method to be the most realistic. Figure 4 is an example of our user study questionnaire. The results suggest that images generated by our method are more likely to be perceived as real photographs. In contrast, Multi-ControlNet, due to significant distortions like color shifts, is perceived as having lower realism. Additionally, the results of RGB↔X [28] seem somewhat specious, with certain structures not adequately preserving the intrinsic features of the images and lacking realism. Moreover, Ctrl-X [14] often drifts away from capturing the essential features of images, which makes them appear less authentic. Due to the separation of material and geometry into two branches for conditional control and the use of a cross-distribution method, our approach achieves more realistic and accurate features.

5. More Comparsion

5.1. Comparison with RGB↔X

We evaluate our model and RGB↔X [28] on untrained indoor and outdoor images, as shown in Figure 2, which include some ground truth intrinsic images, and we use them as conditional inputs in both our method and RGB↔X to generate re-rendered results. These are displayed in the second and fourth columns of Figure 2. Our method demon-

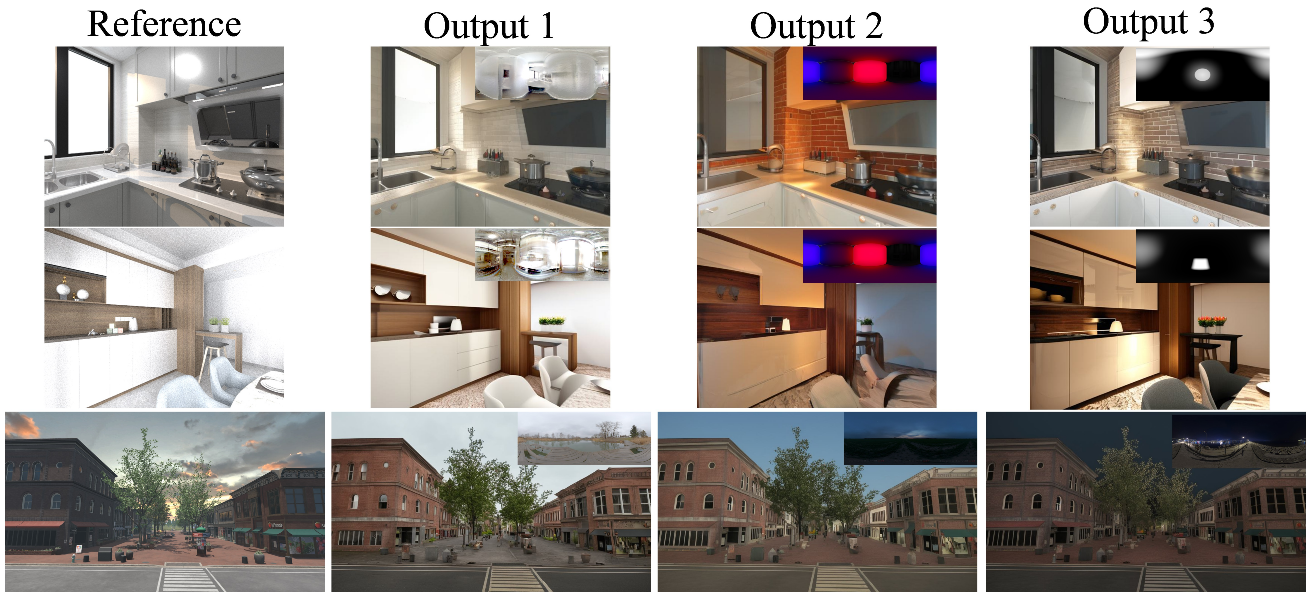
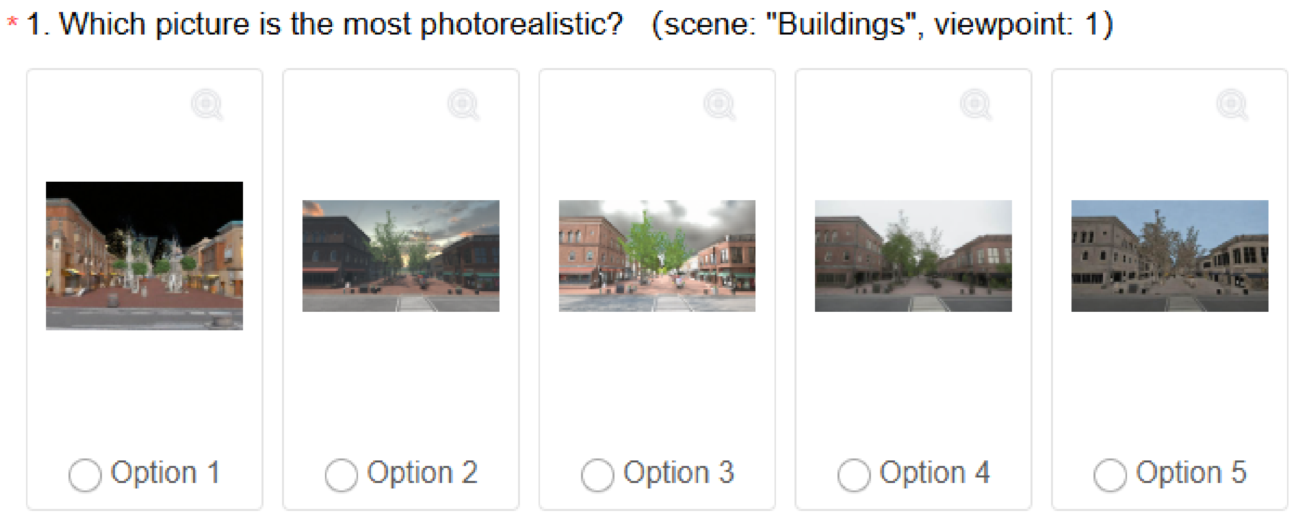


Figure 3. The relighting results of our method. By changing different environment maps as lighting condition, our method can quickly adjust the lighting of the scene to generate realistic relit images.



* 1. Which picture is the most photorealistic? (scene: "Buildings", viewpoint: 1)

◯ Option 1 ◯ Option 2 ◯ Option 3 ◯ Option 4 ◯ Option 5

Figure 4. An example of our user study questionnaire.

strates excellent alignment with the intrinsic image features, closely matching the ground truth. This highlights our method's robustness to different intrinsic image sources and its ability to maintain realism. In contrast, the results of RGB↔X appear chaotic. This may be due to significant differences in the numerical distribution of these intrinsic images compared to those used during the training of RGB↔X. To investigate further, we use the RGB→X process of RGB↔X to regenerate the intrinsic images from the original RGB images and re-render them using the X→RGB process. The results are shown in the third column of Figure 2. Comparing the results of the two RGB↔X approaches with our results, it is evident that the results of RGB↔X contain some errors and lack realism, whereas our results align closely with the ground truth and exhibit a high degree of realism. Additionally, in Table 2 and Table 1 of the main text, we used CLIP and LPIPS metrics to quantitatively compare the results obtained by our method and RGB↔X. The table shows that our method achieves higher CLIP scores and lower LPIPS values across different scene images compared to RGB↔X. This indicates that our results are more photorealistic and have a lower discrepancy

Table 3. Quantitative comparison of different methods and different intrinsic images. 'LP.' is the abbreviation for LPIPS.

Intrinsic	All				A.				M.+R.				A.+N.+D.+S.			
Scores	CLIP ↑	DINO↑	LP. ↓	L2 ↓	CLIP ↑	DINO↑	LP. ↓	L2 ↓	CLIP ↑	DINO↑	LP. ↓	L2 ↓	CLIP ↑	DINO↑	LP. ↓	L2 ↓
GT	0.2733	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ours	**0.3137**	**0.9772**	**0.2569**	**128.1**	**0.3011**	**0.9763**	**0.2576**	**136.7**	**0.3048**	**0.9670**	**0.3162**	**148.6**	**0.3088**	**0.9771**	**0.2590**	**133.3**
Multi.	0.2891	0.9531	0.2910	151.5	0.2984	0.9284	0.5153	185.0	0.2960	0.9603	0.3369	180.5	0.3037	0.9766	0.2621	149.2

from the ground truth.

5.2. Comparison with Multi-ControlNet

We compare our method with the Multi-ControlNet approach. To ensure a fair comparison, we train a separate ControlNet [29] for each intrinsic image type on our entire mixed dataset. Then, We separately use our model and the Multi-ControlNet approach to generate images with various combinations of condition inputs. The comparison results are shown in Figure 11 and Table 3. The $L2$ metric from Meng et al. [16] quantifies faithfulness by calculating the $L2$ distance over all pixels between the guide and output, normalized to [0,1]. Additionally, we can observe that using certain subsets of intrinsic images produces similar results, a discussion that we will delve into in Section 8.

Table 2 and Table 3 present a quantitative comparison of our method with Multi-ControlNet using CLIP, LPIPS, DINOv2 similarity distance, and L2 distance metrics. Our results achieve higher CLIP scores and DINOv2 similarity, lower LPIPS and L2 values. This indicates that our method achieves better alignment with the ground truth compared to Multi-ControlNet when using combinations of different intrinsic images as inputs.

5.3. Comparison with Graphics Engines

As mentioned in Section 1 and Section 4, we compared our method to several popular graphics engines using the CLIP score. Since the calculation of the CLIP score depends on the provided text prompt, we test a wider variety of prompts here to provide more comprehensive quantitative comparison results. As shown in the Table 4, when we used 'engine rendering style' as the text prompt, the results generated by various graphics engines achieved higher CLIP scores. Conversely, when we used 'photorealism' as the text prompt, our results achieved better scores.

5.4. Comparison with Relighting and Editing Methods

Our work focuses on RGB synthesis from intrinsics (more akin to RGB↔X, Ctrl-X, and concurrent PRISM [6] (not open-source yet)). Notably, these approaches do not compare the additional baselines (Neural Gaffer [10], IC-Light [30], and DiLightNet [27]) because of scope and application differences: light-only control vs. intrinsic controls. However, since our method is fully capable of performing relighting tasks, we conducted additional experiments focused specifically on relighting and other instruction-guided editing tasks, with the results presented in the accompanying Figure 6 and Table 5. While the goal of intrinsic control is controllable and photo-realistic generation aligned with input intrinsic, IC-Light, DiLightNet, and Neural Gaffer fail in scene-level application, e.g., with missing shadows and geometry/material errors. For scene editing and image-prompt generation, our design of intrinsic control offers precision while MagicBrush/MGIE/IP-Adapter/OminiControl falter. This demonstrates our superior intrinsic-guided control and cross-distribution realism over the additional baselines.

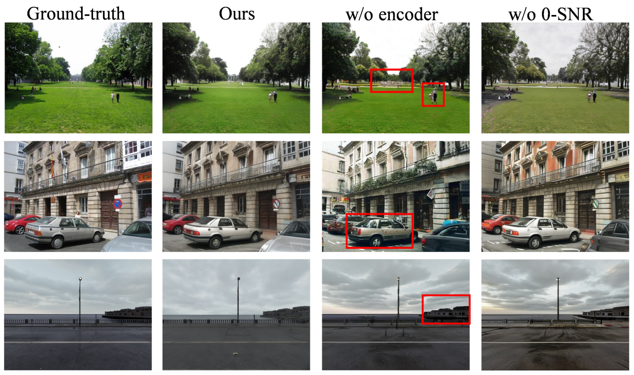


Figure 5. Ablation study on image encoder and 0-SNR strategy.

6. Ablation Study

Effect of cross-distribution training data. To evaluate the impact of cross-distribution training data on the generated results, we trained our framework separately on synthetic data and real data and show the results in Figure 6 in the main text. As shown in the third column of Figure 6 in the main text, the model trained without real data tends to produce overall darker images. While the geometry is maintained, the results look highly unrealistic, with harsh highlights and a noticeable loss of realism compared to our method. In contrast, when training solely on real data, the model generates more realistic images, but many geometric details in the scene are either incorrect or blurred. For instance, in the first row of the fourth column in the im-

	Building(UE5)			Kitchen&Park(Falcor)			Alley(blender)		
Prompt	a ↑	a+b ↑	a+c ↓	a ↑	a+b ↑	a+c ↓	a ↑	a+b ↑	a+c ↓
Reference	0.2629	0.2529	0.2798	0.2651	0.2630	0.2850	0.2508	0.2801	0.2747
Ours	**0.2773**	**0.2703**	**0.2717**	**0.2836**	**0.2822**	**0.2779**	**0.2997**	**0.3036**	**0.2611**

Table 4. Comparison of the results of our method and different graphic engines.
a: BLIP prompt; b: "photorealism"; c:"engine rendering style".

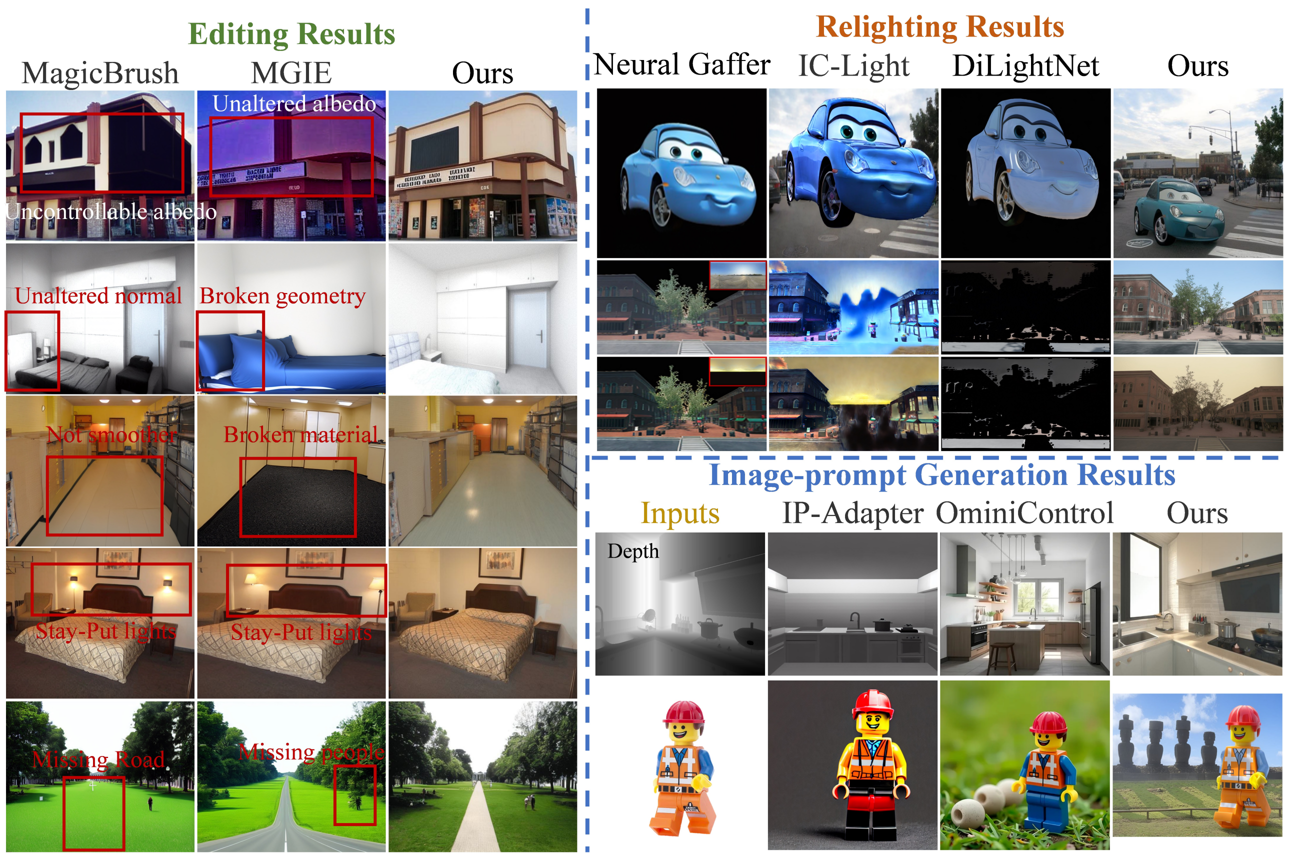


Figure 6. More comparison results with relighting and editing methods.

age, the person at the end of the corridor is missing in the model trained solely on the real dataset. Additionally, the results obtained from this ablation method generally exhibit a warmer color tone. Only our approach enables the generated images to achieve high realism while preserving precise, controllable geometry.

Effect of the separate design of material and geometry. In the previous section, we explained the necessity of using a mixed dataset of real and synthetic data. However, simply expanding the dataset is not sufficient, as it can lead to results with geometric color confusion. For example, in the first row of the fifth column, the person's clothing has been transformed into grass. In the second row, there are prominent color bleeding errors affecting the items on the kitchen countertop and the window. To address the cross-distribution training challenge, we divided the input intrinsic images into two groups and used two separate branches to manage each group. To validate the effectiveness of this design, we train a model on the same dataset as ours but with a single branch. In this model, all intrinsic images pass through the same ControlNet to control the latent diffusion model. By separating the geometry and material branches, the results, as shown in the second column of Figure 6 in the main text, are better aligned with the intrinsic image features.

Effect of lighting control. We compared the results of incorporating lighting as a conditional input versus omitting it, as depicted in the last column of Figure 6 in the main

Applications	Relighting			Scene editing		Image-prompt		All
Methods	Neural Gaffer	IC-Light	DiLightNet	MagicBrush	MGIE	IP-Adapter	OmniControl	Ours
CLIP ↑	0.254	0.271	0.244	0.281	0.250	0.272	0.284	**0.298**
SSIM ↑	0.136	0.332	0.064	-	-	0.277	0.310	**0.665**

Table 5. More quantitative comparison with relighting and editing methods.

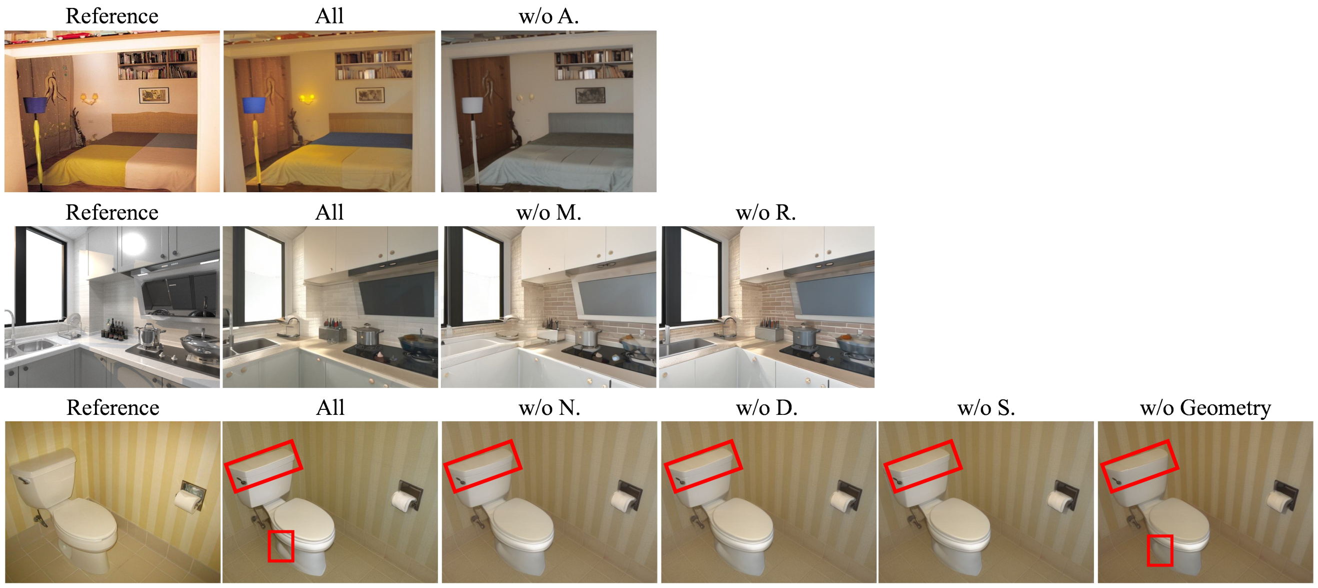


Figure 7. Impact of different intrinsic images, including albedo (A.), roughness (R.), metallicity (M.), normal (N.), depth (D.), and semantic segmentation (S.). 'All' indicates the results generated using all intrinsic images as conditions, including lighting. 'w/o X' represents the results obtained using all intrinsic images as conditions except for X. 'Geometry' means all the geometry intrinsic images, including normal, depth, and semantic segmentation.

text. Without lighting conditions, the generated images display random lighting effects, as seen with the reflections on the floor in the first row of the last column of Figure 6 in the main text. In enclosed indoor spaces with windows, shown in the second row of the last column, we noticed that without lighting input, the windows appear black or gray, causing the room to be generally dim. However, when lighting is included, the windows allow light to enter, brightening the room. This demonstrates that our lighting condition plays a crucial role, enabling the generated results to more accurately reflect the lighting information of the provided environment map.

Additionally, we provided more generated results after changing the panoramic environment maps, as shown in Figure 3. It is evident that the lighting from different environment maps is reflected in the resulting images. For example, in the first two rows with 'Output 2', the middle section of the result images is clearly illuminated with red light, while the sides display some blue light. This demonstrates that our lighting control effectively influences the images, making it a useful tool for altering the overall lighting of scene images.

Image encoder. We compare the generated results obtained with and without using the condition image encoder mentioned in Section 3.2 in the main text and show the results in Figure 5. With the original image encoder from ControlNet [29], there are clear geometric and material inaccuracies compared to the ground truth, as shown in the third column. However, our results in the second column preserve the local geometry present in the ground truth. The experiment shows that our improved image encoder has a clear advantage in extracting fine geometry from intrinsic images, allowing our network to learn more precise control over the details in various images.

Zero terminal SNR strategy. As shown in Figure 5, the results without using zero SNR strategy in the last column have a significant color discrepancy compared to the ground truth in the first column, appearing overall darker and redder, whereas the results using the zero terminal SNR strategy (Ours) show almost no color deviation. The zero termi-

Table 6. Quantitative comparison of object insertion and scene editing tasks in different methods.

Application	Object insertion			Scene editing	
Methods	RGB addition	Ours	AnyDoor	Ours	InstructPix2Pix
CLIP ↑	0.282	**0.296**	0.289	**0.299**	0.282
DINOv2-selfsim ↑	0.6716	**0.6748**	0.6683	-	-
DINOv2-sim ↑	-	-	-	**0.9794**	0.9031

nal SNR strategy ensures consistency in the diffusion process between training and inference by fully adding noise during training. This closely matches the initial step of the inference process, resulting in generated images that align with the original data distribution.

7. Quantitative Results of Object Injection and Scene Editing

In applications focused on object insertion and scene editing, we conducted a quantitative comparison of our results against those produced by direct RGB insertion and editing, RGB↔X [28], AnyDoor [5], and InstructPix2Pix [4], as shown in Table 6. Notably, in the object insertion application, we utilize only albedo and normal as conditional inputs to minimize interference from other unchanged conditions. Similarly, in the scene editing section, we employ only albedo and the edited intrinsic image as conditional inputs. Figure 7 and Figure 9 in the main text and Table 6 show that our method demonstrates a more precise alignment with the changes in the conditional images and achieves higher CLIP scores and DINOv2 similarity on both object insertion and scene editing applications.

Specifically, we use the DINOv2 self-similarity distance instead of DINOv2 similarity distance to measure the quality of the generated results for object insertion. Because the background image and the inserted object originate from two different images. Simply inserting the object at the RGB level can result in significant discrepancies, including differences in tone, highlights, and shadows. The self-similarity distance helps evaluate whether the result of the object insertion appears as a cohesive image, specifically whether the inserted object aligns with the lighting and shadow effects of the background. As shown in Table 6, compared to direct insertion at the RGB image level and AnyDoor [5], our approach achieves higher image self-similarity values. This indicates that our method allows the inserted object to blend more seamlessly with the background features.

8. Discussion on the Impact of Different Intrinsic Images

During the experiment, we found that different combinations of intrinsic images as conditional inputs might result in similar generated images. This inspired us to explore which conditions play a decisive role in the generation results and which conditions are relatively redundant in the generation process. Figure 6 in the main text and Figure 3 has demonstrated that the environment map influences the final result as a lighting condition. Therefore, we will now examine how the other six intrinsic images (albedo, normal, depth, metallicity, roughness, and semantic segmentation) affect the generated outcome with the same lighting. The first row in Figure 7 illustrates that albedo is a crucial conditional input for preserving the overall perception of the reference image. Without using albedo as a condition, the colors in the generated image would appear random.

The second row of Figure 7 shows the results generated by our model in the absence of metallicity or roughness conditions. The results indicate that for some metallic and smooth materials, including metallicity helps maintain better alignment with the reference image compared to when metallicity is not included. For instance, the black section above the stovetop is rendered as silver in the absence of metallicity. This demonstrates that both metallicity and roughness influence the generated results, but roughness is somewhat redundant compared to metallicity.

The third row of Figure 7 presents the results generated by our model in the absence of normal, depth, or semantic segmentation as conditional inputs. As shown in Figure 7, incorporating one or more geometric intrinsic images as conditions, especially normal, can improve the geometric accuracy of the generated results compared to using only albedo. For instance, when normals are included, the edges and indentations at the base of the toilet are rendered more clearly compared to when normals are not included. Therefore, by adding geometric intrinsic images, these geometric inaccuracies are appropriately corrected.

9. More Details of Network Architecture

9.1. Structure and Components

We show the details of training and inference network architecture in the Figure 9 and Figure 10 respectively. We apply cross-feature injection in both branches during training process, as shown in Figure 9. Specifically, at each training step t, geometry features f_g^S from the geometry branch are injected into the material branch when the input data is synthetic, while material features f_m^T are injected into the geometry branch when the input data real-world data, both

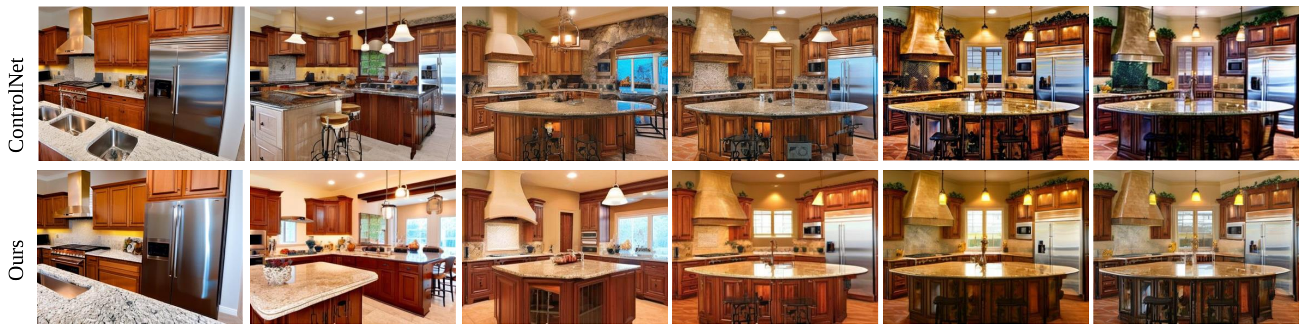


Figure 8. Comparison of the convergence between ControlNet [29] and our framework, where ControlNet uses a single branch to control both material and geometry generation.

at the l-th layer of the latent diffusion model. At each inference time step t, we only use the material branch for image generation with geometry feature injection at the l layer, as shown in Figure 10.

The realistic discriminator is a neural network model designed for distinguishing between target and generated images while also classifying the images into different distributions. It consists of a series of shared convolutional layers followed by two separate heads for multitask learning. The shared layers begin with a 2D convolutional layer that takes a 4-channel input and applies a 64-filter convolution with a kernel size of 4, a stride of 2, and a padding of 1, followed by a LeakyReLU activation function. This is followed by another convolutional layer with 128 filters, batch normalization, and LeakyReLU activation. The pattern continues with 256 and 512 filters in subsequent layers, each followed by batch normalization and LeakyReLU. After flattening the output, the network branches into two heads. The target/generated head consists of a single linear layer that outputs a single value, indicating whether the input image is target or generated. The distribution classification head also consists of a single linear layer, but it outputs a vector with a length equal to the number of distributions, providing a classification score for each distribution. This architecture allows the discriminator to perform both target/generated discrimination and distribution classification simultaneously. This approach ensures that the geometry in the image generated by the material branch aligns with the geometry predicted by the geometry branch while also preserving the realistic features of the material branch.

9.2. Design Rationale

Rationale for real data lighting pair generation. Our approach in Sec 3.3 of the main text is a compromise on the scarcity of matched real-world RGB image with corresponding HDR map. Our design of decoupling geometry and material control allows us to effectively harness the photo-realism inherent in real-world data and the controllable accuracy of synthetic data. The extensive synthetic data in the training process provides precise pairwise data consisting of RGB images and HDR maps, which effectively eliminates negative impact from the inaccuracies found in real-world data, as confirmed by Fig 6 in the main text.

Effect of GAN loss. We aim to generate photo-realistic images from synthetic intrinsics. The lack of paired synthetic and real data prevents explicit definition of the two differing image distributions, motivating us to achieve robust cross-distribution generation. In contrast to existing diffusion-based relighting methods, which use paired data and explicit targets in training to simplify learning distributional transformations, we employ GAN loss to implicitly disentangle the distributions between real-world and synthetic images adversarially. This allows our model's two branches to capture distinct features of each distribution and successfully bridge this gap, facilitating effective cross-distribution generation from unpaired sources.

10. Convergence Speed of Geometry and Material

Figure 8 shows the intermediate results during the training process for both ControlNet [29] and our model. We attempt to use a single branch to control both the geometry and material of the generated images using ControlNet. During the training process of ControlNet, we can observe that the convergence speeds of material and geometry differ. As shown in Figure 8, the geometry tends to converge much faster than material. At an early training stage, the geometric features of the generated results are already globally well-controlled, but the color and style are still not precisely managed, with noticeable color errors in the details. As the training progresses, the geometry continues to refine locally, but the slow convergence of material causes the color and style to lag behind the ground truth. Since geometric control has been achieved, the network as a whole tends to stabilize. This results in the final generated images having issues like being too dark or too magenta. To avoid this issue, we separated the material and geometry features by using two ControlNet branches with non-shared parameters as mentioned in Section 3.4 in the main text, which prevents interference between the optimization of material and geometry.

11. Visualization of Entire Intrinsic Images

We show the entire intrinsic image input achieved from real-world photos in Figure 13, and compare the generated images of our methods with the original photos in 1.

12. Limitation and Failed Cases

Our model is capable of generating RGB images with the desired features from intrinsic images in most cases, but it does exhibit some limitations and failure cases. For real-world images, where accurate intrinsic images are not available, we generate them using the method outlined in IntrinsicImageDiffusion [12]. If the normal or depth maps produced by this method lose geometric information, as illustrated in the first two rows of Figure 12, our results may have overall colors similar to the ground truth but display disorganized local geometry. Fortunately, this component can be updated and replaced with a more advanced model to produce more accurate intrinsic images. Additionally, the last two rows of Figure 12 demonstrate that when adjacent colors in the albedo map are very similar, our model struggles to distinguish between them, resulting in the generated output being rendered as a single color.

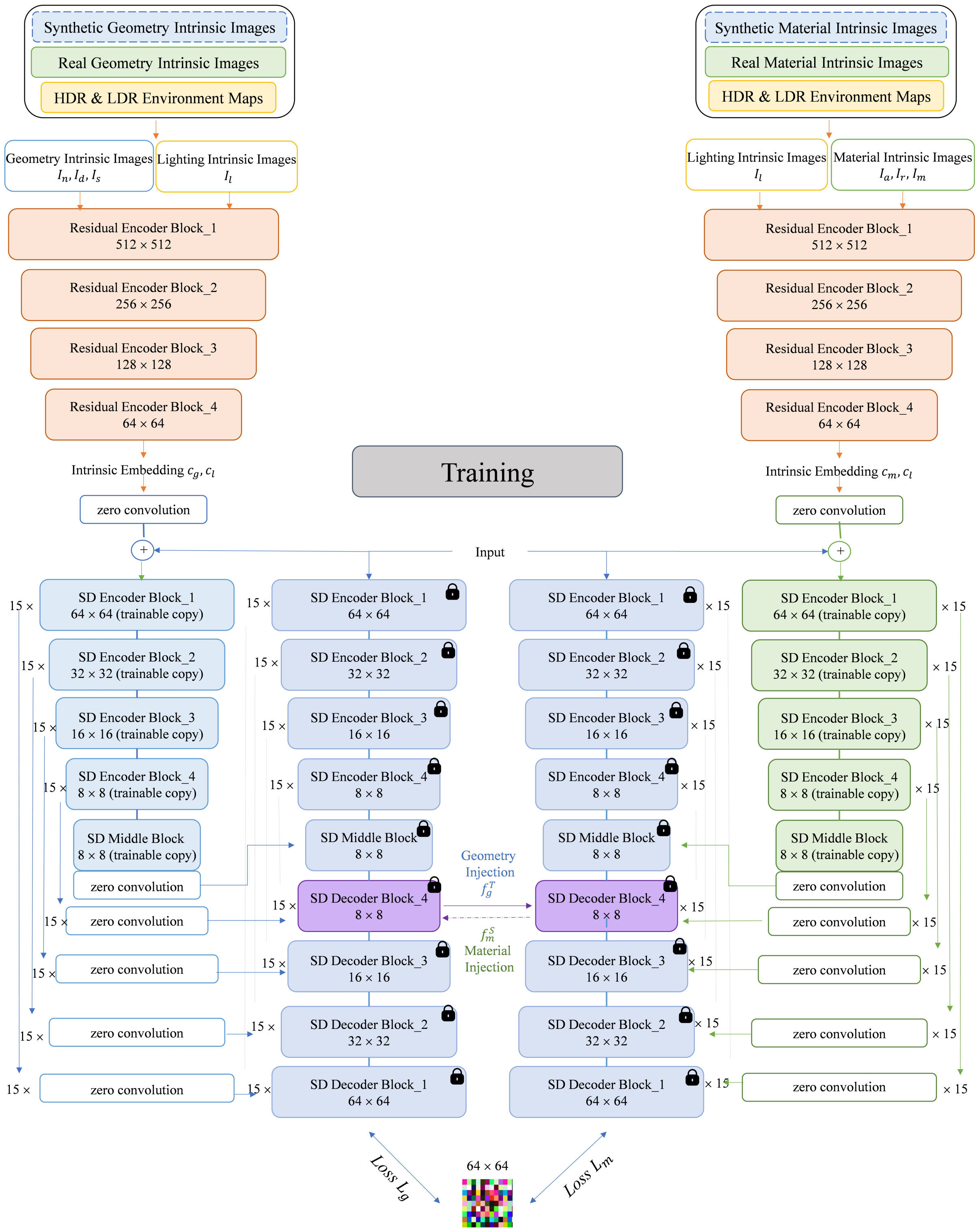

Synthetic Geometry Intrinsic Images
Real Geometry Intrinsic Images
HDR & LDR Environment Maps
Geometry Intrinsic Images I_n, I_d, I_s
Lighting Intrinsic Images I_l
Residual Encoder Block_1 512 × 512
Residual Encoder Block_2 256 × 256
Residual Encoder Block_3 128 × 128
Residual Encoder Block_4 64 × 64
Intrinsic Embedding c_g, c_l
zero convolution
Synthetic Material Intrinsic Images
Real Material Intrinsic Images
HDR & LDR Environment Maps
Lighting Intrinsic Images I_l
Material Intrinsic Images I_a, I_r, I_m
Intrinsic Embedding c_m, c_l
Training
Input
SD Encoder Block_1 64 × 64 (trainable copy)
SD Encoder Block_2 32 × 32 (trainable copy)
SD Encoder Block_3 16 × 16 (trainable copy)
SD Encoder Block_4 8 × 8 (trainable copy)
SD Middle Block 8 × 8 (trainable copy)
SD Encoder Block_1 64 × 64
SD Encoder Block_2 32 × 32
SD Encoder Block_3 16 × 16
SD Encoder Block_4 8 × 8
SD Middle Block 8 × 8
SD Decoder Block_4 8 × 8
SD Decoder Block_3 16 × 16
SD Decoder Block_2 32 × 32
SD Decoder Block_1 64 × 64
15 ×
× 15
Geometry Injection f_g^T
f_m^S Material Injection
Loss L_g
Loss L_m
64 × 64

Figure 9. Details of feature injection in our training network architecture. The dashed line represents synthetic data streams, and the solid line represents real data streams. When synthetic data is used as input, it is processed through the ControlNet of both the geometry and material branches. Material branch features are injected into the geometry branch, and the prediction is derived from the geometry branch to compute the loss with Gaussian noise. For real data, the process is similar, but geometry branch features are injected into the material branch, and the prediction is derived from the material branch. Lighting information is concatenated into the conditions of both branches, controlling both branches along with the other conditions. During training, losses from both branches are combined for joint optimization.

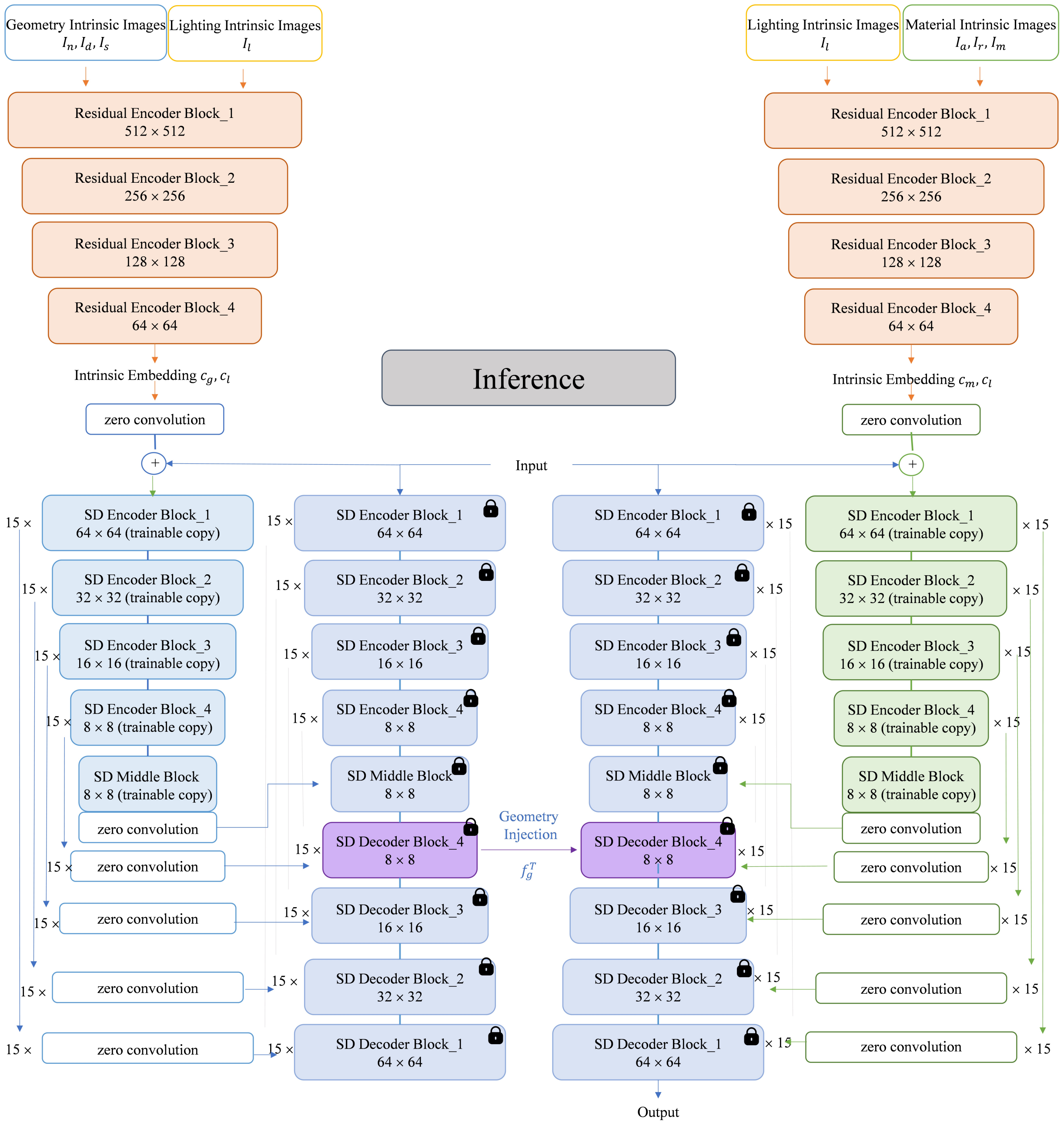

Geometry Intrinsic Images I_n, I_d, I_s
Lighting Intrinsic Images I_l
Lighting Intrinsic Images I_l
Material Intrinsic Images I_a, I_r, I_m
Residual Encoder Block_1 512 × 512
Residual Encoder Block_2 256 × 256
Residual Encoder Block_3 128 × 128
Residual Encoder Block_4 64 × 64
Intrinsic Embedding c_g, c_l
Intrinsic Embedding c_m, c_l
Inference
zero convolution
Input
SD Encoder Block_1 64 × 64 (trainable copy)
SD Encoder Block_2 32 × 32 (trainable copy)
SD Encoder Block_3 16 × 16 (trainable copy)
SD Encoder Block_4 8 × 8 (trainable copy)
SD Middle Block 8 × 8 (trainable copy)
SD Encoder Block_1 64 × 64
SD Encoder Block_2 32 × 32
SD Encoder Block_3 16 × 16
SD Encoder Block_4 8 × 8
SD Middle Block 8 × 8
SD Decoder Block_4 8 × 8
SD Decoder Block_3 16 × 16
SD Decoder Block_2 32 × 32
SD Decoder Block_1 64 × 64
Geometry Injection
f_g^T
15 ×
× 15
Output

Figure 10. Details of our inference network architecture. During inference, regardless of whether the input data is synthetic or real, the geometric conditions are processed through the geometry branch's ControlNet, and the material conditions are processed through the material branch's ControlNet. Lighting information is concatenated into the conditions of both branches, controlling both branches along with the other conditions. The features from the geometry branch are injected into the material branch, which then generates the final output image.

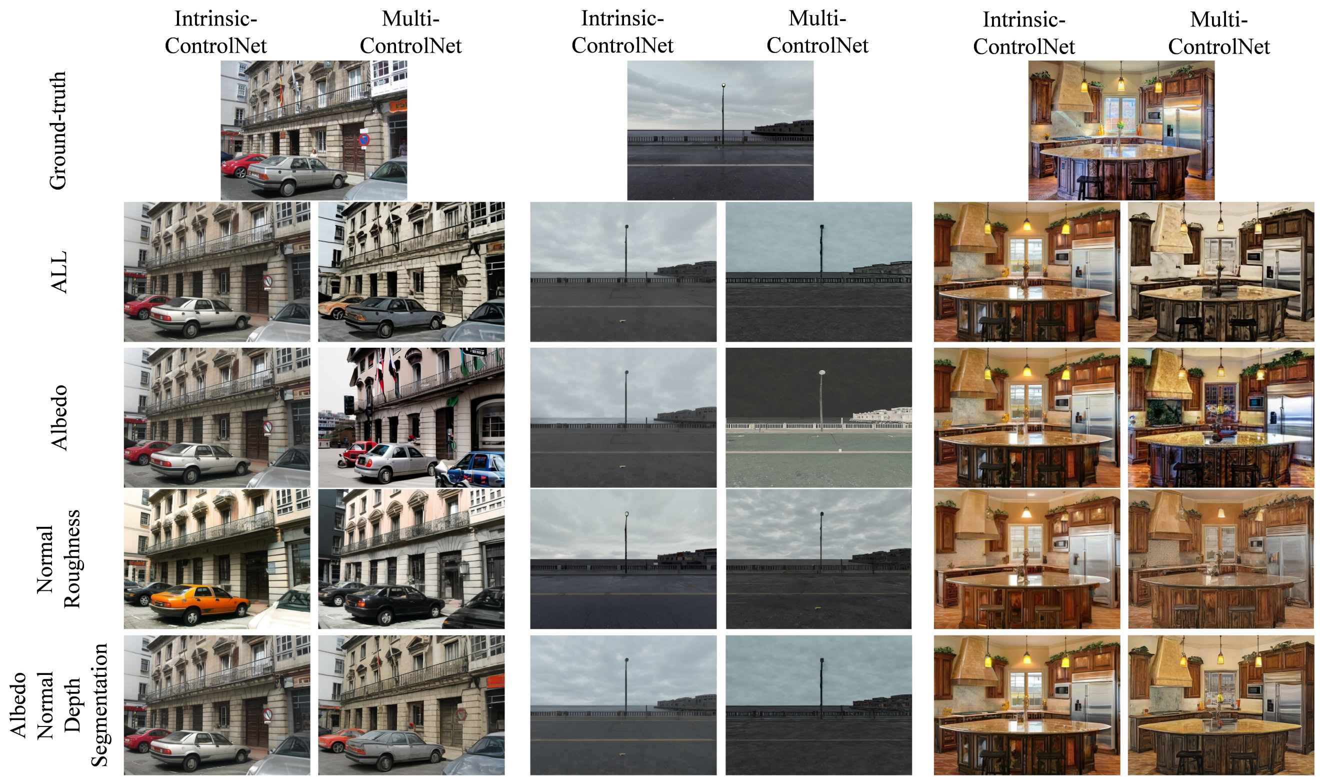


Figure 11. Results of different combination of intrinsic conditions between IntrinsicControlNet and Multi-ControlNet.

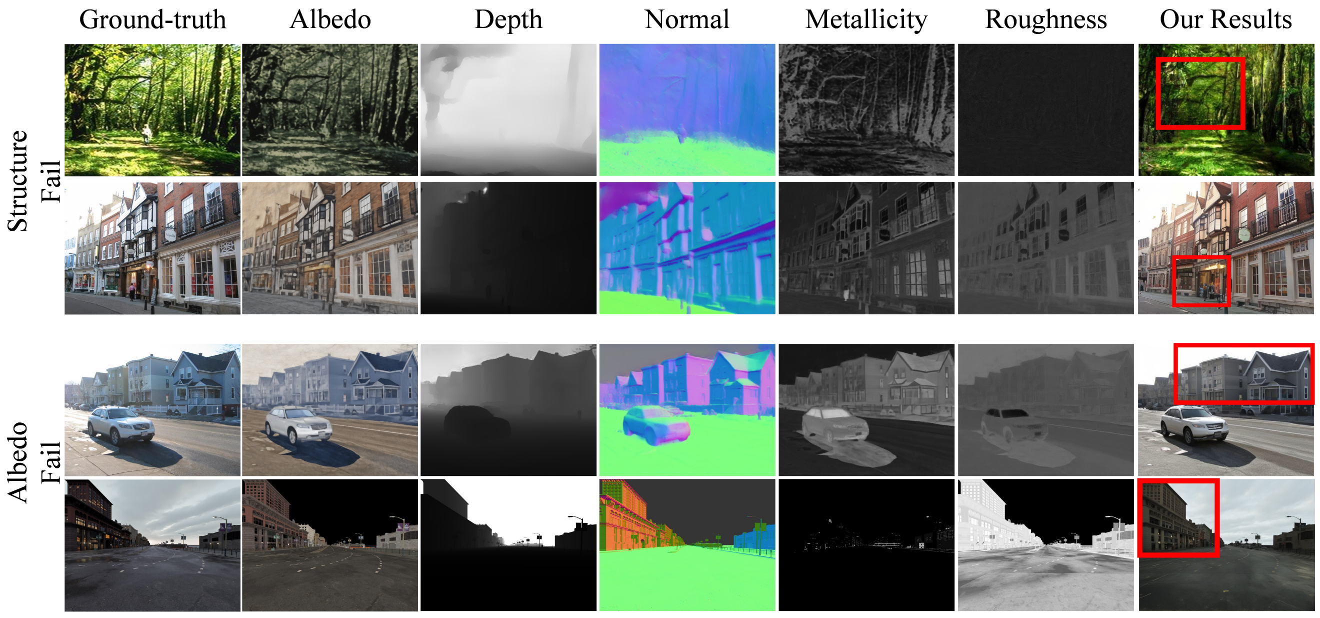


Figure 12. Some failure cases of our framework.

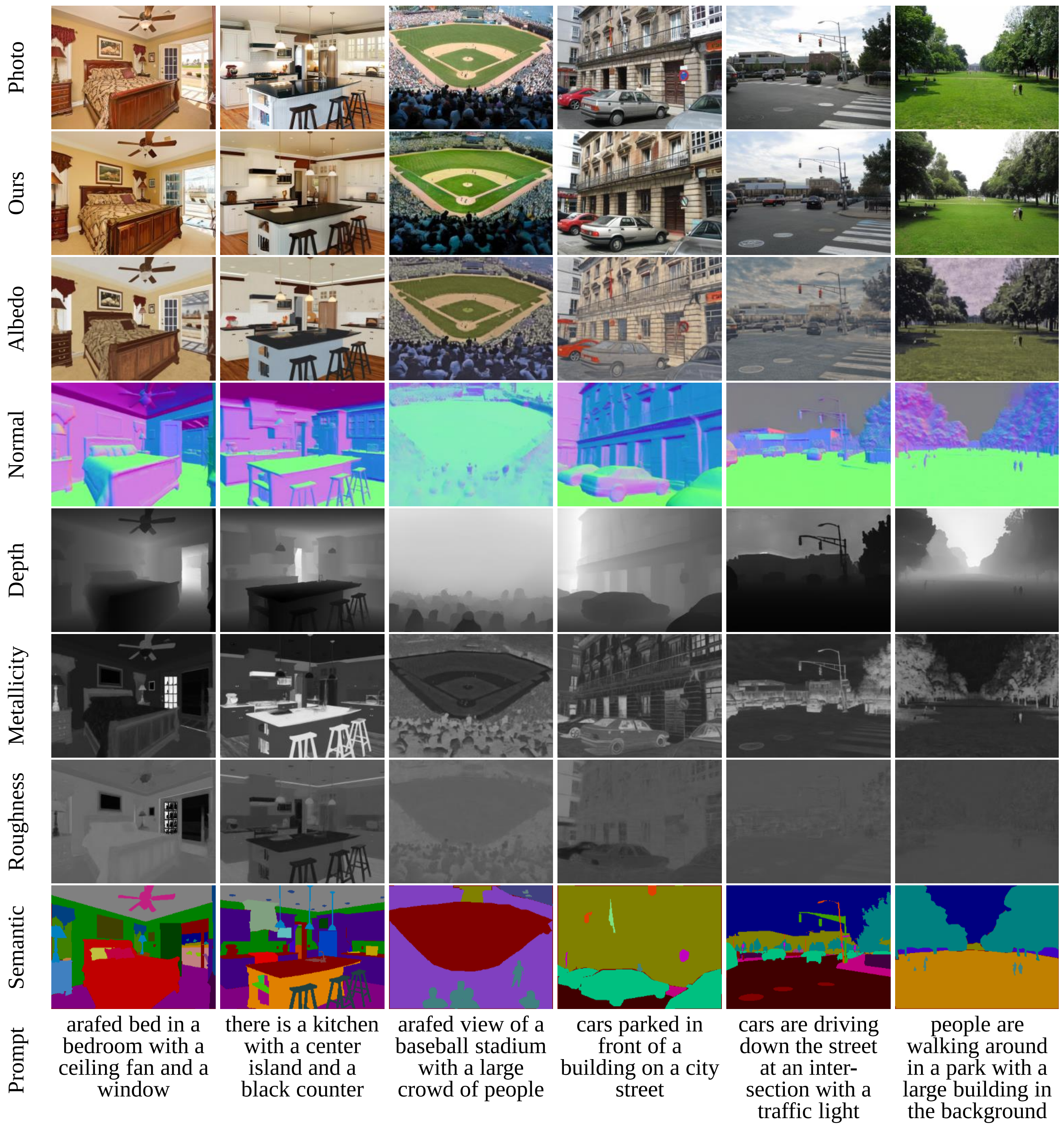


Photo
Ours
Albedo
Normal
Depth
Metallicity
Roughness
Semantic
Prompt
arafed bed in a bedroom with a ceiling fan and a window
there is a kitchen with a center island and a black counter
arafed view of a baseball stadium with a large crowd of people
cars parked in front of a building on a city street
cars are driving down the street at an inter-section with a traffic light
people are walking around in a park with a large building in the background

Figure 13. Our framework uses predicted multiple intrinsic images to generate realistic images that resemble the original. Here we show all intrinsic images with the results.

References

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 2

[2] N. Benty et al. The Falcor Rendering Framework. `https://github.com/NVIDIAGameWorks/Falcor`. Accessed: Sep. 29, 2024. 1, 3

[3] Blender Foundation. Blender - a 3D modelling and rendering package. `https://www.blender.org/`. Accessed: Sep. 29, 2024. 1, 3

[4] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 8

[5] Xi Chen, Lianghua Huang, Yu Liu, Yujun Shen, Deli Zhao, and Hengshuang Zhao. Anydoor: Zero-shot object-level image customization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6593–6602, 2024. 8

[6] Alara Dirik, Tuanfeng Wang, Duygu Ceylan, Stefanos Zafeiriou, and Anna Frühstück. Prism: A unified framework for photorealistic reconstruction and intrinsic scene modeling. 2025. 5

[7] Epic Games. Unreal Engine. `https://www.unrealengine.com/`. Accessed: Sep. 29, 2024. 1, 3

[8] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *ArXiv*, abs/2104.08718, 2021. 1

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *ArXiv*, abs/1706.08500, 2017. 2

[10] 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 *Advances in Neural Information Processing Systems*, 2024. 5

[11] Joni MercadoS J Bennett and others. Poly Haven. `https://polyhaven.com/hdris/`. Accessed: Nov. 13, 2024. 2

[12] Peter Kocsis, Vincent Sitzmann, and Matthias Nießner. Intrinsic image diffusion for single-view material estimation. *arXiv preprint arXiv:2312.12274*, 2023. 2, 10

[13] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 1

[14] Kuan Heng Lin, Sicheng Mo, Ben Klingher, Fangzhou Mu, and Bolei Zhou. Ctrl-x: Controlling structure and appearance for text-to-image generation without guidance. *arXiv preprint arXiv:2406.07540*, 2024. 2, 3

[15] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[16] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021. 5

[17] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. 1

[18] Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9914–9925, 2024. 2

[19] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1

[20] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ArXiv*, abs/2102.12092, 2021. 1

[21] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 102–118. Springer, 2016. 2

[22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1

[23] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1

[24] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023. 2

[25] Guangcong Wang, Yinuo Yang, Chen Change Loy, and Ziwei Liu. Stylelight: Hdr panorama generation for lighting estimation and editing. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[26] Jiazhi Yang, Shenyuan Gao, Yihang Qiu, Li Chen, Tianyu Li, Bo Dai, Kashyap Chitta, Penghao Wu, Jia Zeng, Ping Luo, Jun Zhang, Andreas Geiger, Yu Qiao, and Hongyang Li. Generalized predictive model for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[27] 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 2024 Conference Papers*, 2024. 5

[28] Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. Rgb$\leftrightarrow$x: Image decomposition and synthesis using material-and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2, 3, 8

[29] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 5, 7, 9

[30] 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 *The Thirteenth International Conference on Learning Representations*, 2025. 5

[31] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. 1

[32] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In *Computer Vision – ECCV 2014*, pages 668–686, Cham, 2014. Springer International Publishing. 2

[33] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 2

[34] Jingsen Zhu, Fujun Luan, Yuchi Huo, Zihao Lin, Zhihua Zhong, Dianbing Xi, Rui Wang, Hujun Bao, Jiaxiang Zheng, and Rui Tang. Learning-based inverse rendering of complex indoor scenes with differentiable monte carlo raytracing. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–8, 2022. 2