IntrinsicWeather: Controllable Weather Editing in Intrinsic Space

Supplementary Material

A. Related works

Inverse rendering. Inverse rendering aims to recover scene-level physical properties such as geometry, material, and lighting from observed images. Intrinsic image decomposition is a common formulation of inverse rendering, and our method focuses on it, so we mainly review intrinsic image decomposition in this section. Early studies [1, 3, 5] predominantly relied on hypothesis priors derived from Retinex theory [7, 14]. Recent intrinsic image decomposition methods [9, 23, 27, 30, 32] often used a learning-based framework on large-scale dataset [15, 24, 32]. These deep learning-driven approaches have enabled intrinsic image decomposition models to surpass the limitations of Lambertian reflectance assumptions, achieving predictive capacities for physically based rendering-oriented material properties, geometric structures, and illumination parameters.

B. Preliminary: diffusion model for rendering

Given image $\boldsymbol{I} \in \mathbb{R}^{H\times W\times 3}$ and intrinsic map $\boldsymbol{y} \in \mathbb{R}^{H\times W\times C}$, diffusion model employs a pre-trained encoder to map them from pixel space to a latent space and then get the latent variable:

$$\boldsymbol{x}_0 = \mathcal{E}(\boldsymbol{I}), \quad \boldsymbol{z}_0 = \mathcal{E}(\boldsymbol{y}). \tag{1}$$

Following Flow Matching [19], a random noise is added to the latent variable:

$$\boldsymbol{z}_t = (1-t)\boldsymbol{z}_0 + t\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2}$$

where t represents denoising timestep and $\boldsymbol{\epsilon}$ is random noise sampled from a Gaussian distribution. Then DiT [22] is used to estimate the velocity field at a specific timestep. This is achieved by minimizing the following loss function:

$$\mathcal{L}_\theta = \mathbb{E}_{t\sim\mathcal{U}(0,1),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\|\boldsymbol{v}_\theta(\boldsymbol{z}_t, \boldsymbol{c}, t) - (\boldsymbol{\epsilon} - \boldsymbol{z}_0)\|_2^2]. \tag{3}$$

Once the velocity field is estimated, the noisy latent variable is progressively denoised step by step, resulting in a clean estimation of the original latent variable.

C. Implementation details

We repurpose two separate Stable Diffusion 3.5 Medium (SD3.5)[1] to enable the inverse and forward renderers following Sec. B. In this section, we will introduce the implementation details of the inverse and forward renderers, IMAA, CLIP-interpolation, along with the training strategy.

[1] https://huggingface.co/stabilityai/stable-diffusion-3.5-medium

C.1. Inverse renderer

VAE [11] of SD3.5 encodes the image into a 16-channel latent space. Following Instruct-Pix2pix [2], we concatenate the latent original image and diffusion noise together, and the final input channel size is 32. We notice that the intrinsic maps remain consistent when solely supervised by data, so unlike previous works such as Wonder3D [20], we do not introduce an explicit mechanism for information exchange among maps.

C.2. Forward renderer

Following Instruct-Pix2pix [2], we concatenate the set of intrinsic maps and diffusion noise together, and the final input channel size is 96. Furthermore, to enhance consistency with the intrinsic maps while maintaining generative quality, we apply classifier-free guidance. When training, we drop intrinsic maps randomly following RGB↔X [31], and we set the drop probability $p = 0.1$. In inference, we set the guidance scale as 7.5 and the image guidance scale as 3.

C.3. Intrinsic Map-Aware Attention (IMAA)

The goal of the IMAA is to generate map-specific spatial attention masks that guide the cross-modal attention between image features and textual tokens. We present a network architecture in Fig. 1. It takes patch-level features from a frozen DINO backbone and combines them with a learnable embedding for each map type, projecting both into a common feature space. The two streams are fused by lightweight convolutional layers, and a convolutional head outputs a single-channel gating map. After upsampling and sigmoid normalization, this gating map is scaled and inserted into the attention bias, allowing image tokens to be weighted adaptively according to the target map.

C.4. Heuristic-guided progressive training for IMAA.

We find that the model is difficult to converge if we finetune the DiT [22] with randomly initialized IMAA directly. This is because IMAA provides auxiliary guidance for DiT. In the early period of finetuning, the model does not have sufficient capacity to generate plausible estimation, so the loss cannot provide enough information to help IMAA optimize. To this end, we propose heuristic-guided progressive training for IMAA.

Firstly, we pretrain IMAA heuristically. Following the observation mentioned in the main paper, we design a simple yet effective heuristic to generate importance masks for various intrinsic maps according to the ground-truth, iden-

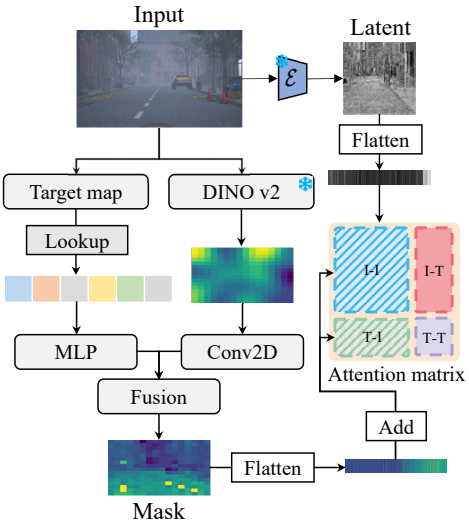

Figure 1. Network architecture of intrinsic map-aware attention.

tifying the most visually or physically significant regions in a scene. The detailed design is as follows.

- **Albedo.** Brighter regions are emphasized, as they are perceptually more salient.
- **Normal.** Areas with sharp geometric changes or large homogeneous surfaces are highlighted.
- **Roughness.** Smooth surfaces (low roughness) are prioritized, since they dominate reflections.
- **Metallicity.** Metallic regions are emphasized due to their unique appearance.
- **Irradiance.** Both brightly lit areas and shadow boundaries are emphasized for lighting consistency.

After obtaining masks m for different maps, we use a supervised learning paradigm to train IMAA, using the following loss function:

$$\mathcal{L}_{\text{IMAA}} = \lambda_1 \cdot \mathcal{L}_{\text{IMAE}} + \lambda_2 \cdot \mathcal{L}_{\text{SSIM}} + \lambda_3 \cdot \mathcal{L}_{\text{grad}}, \quad (4)$$

where $\mathcal{L}_{\text{MSE}}$ provides pixel-level fidelity, , $\mathcal{L}_{\text{SSIM}}$ maintains perceptual similarity, and $\mathcal{L}_{\text{grad}}$ ensures structure preservation. We set $\lambda_1 = 1.0$, $\lambda_2 = 0.5$, $\lambda_3 = 0.1$.

Though we find that applying the pretrained IMAA to the finetuned diffusion model can help improve performance (as shown in Sec. E), we freeze the IMAA and finetune DiT to help it adapt IMAA's guidance better.

C.5. CLIP space interpolation

We leverage a linear interpolation to achieve fine-grained weather control. A naive approach is to directly interpolate between two weather prompts, such as "An overcast day" and "A rainy day". However, this approach results in uncontrollable generation results. Thus, we design an interpolation approach based on weather direction, as introduced in the main paper. We conduct a simple experiment on the original Stable Diffusion, as shown in Fig. 2.

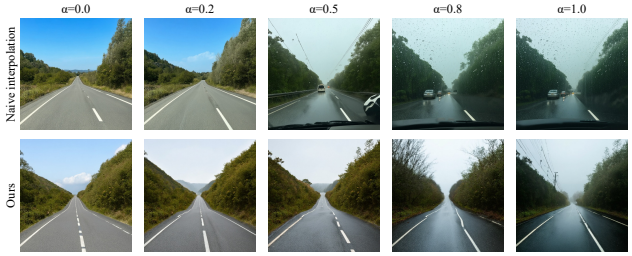

Figure 2. Different interpolation approaches. The naive approach produces uncontrollable results, while our method generates more consistent results.

C.6. Feature distillation of forward renderer

To preserve the strong generative prior of the original pre-trained Stable Diffusion model, we employ a feature distillation strategy during the training of our forward renderer. This technique regularizes the student model (our forward renderer), encouraging its internal feature representations to remain aligned with those of a frozen, pre-trained teacher model (SD3.5). This helps enhance the realism when applying CLIP-interpolation to generate fine-grained weather control. The teacher model is the original, frozen SD 3.5 DiT backbone. It does not receive any intrinsic map conditioning. The student model is our forward renderer, which has the same architecture but is fine-tuned to accept intrinsic maps as additional conditions concatenated to the noisy latents. We extract intermediate features from a predefined set of layers L_{distill} from both the student and teacher DiT backbones. Let $f_s^{(l)}$ and $f_t^{(l)}$ be the feature maps from the l-th layer of the student and teacher, respectively, where $l \in L_{\text{distill}}$. To align the features, we apply Layer Normalization (LN) to mitigate scale differences between feature activations. The distillation loss for a single layer l is a combination of Mean Squared Error (MSE) and a cosine similarity-based loss:

$$\begin{aligned}\mathcal{L}_{\text{distill}}^{(l)} = &\frac{1}{2}\mathcal{L}_{\text{MSE}}(\text{LN}(f_s^{(l)}), \text{LN}(f_t^{(l)})) \\ &+ \frac{1}{2}\left(1 - \text{sim}_{\cos}(f_s^{(l)}, f_t^{(l)})\right)\end{aligned} \quad (5)$$

where $\text{sim}_{\cos}$ denotes the cosine similarity. The MSE term enforces a strict, token-wise numerical match, while the cosine similarity term ensures that the features are structurally similar by aligning their vector directions. The total feature distillation loss is the average loss across all selected layers:

$$\mathcal{L}_{\text{distill}} = \frac{1}{|L_{\text{distill}}|} \sum_{l \in L_{\text{distill}}} \mathcal{L}_{\text{distill}}^{(l)} \quad (6)$$

The feature distillation loss is incorporated into the main training objective as a weighted regularization term. The final loss for the forward renderer is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{FM}} + \lambda_{\text{distill}} \mathcal{L}_{\text{distill}} \tag{7}$$

where $\mathcal{L}_{\text{FM}}$ is the Flow matching loss, and λ_{distill} is a hyperparameter that controls the strength of the distillation. In our experiments, we use $L_{\text{distill}} = [5, 10, 15, 20]$ and set λ_{distill} to 0.1.

C.7. Training details

Besides WeatherSynthetic and WeatherReal, we also use InteriorVerse [32] and Hypersim [24] in training our inverse rendering diffusion and forward rendering diffusion. Although there is a distributional difference between indoor datasets and our task scenario, they can still help the model learn the relationship between intrinsic maps and rendered images. For the inverse rendering model, we freeze the VAE and text encoders and train only the DiT and IMAA modules. We first only finetune the Diffusion model. The inverse renderer is optimized using AdamW [21] with a learning rate of 1e-5 and batch size of 24 on each GPU, training for approximately 2K iterations with 512 resolution, costing about 20 hours on eight 4090 GPUs. We then train IMAA for about 5 hours on a single 4090 GPU, with a learning rate of 1e-4 and batch size of 96. Then we resume training the inverse renderer with frozen IMAA for 1K iterations, costing about 10 hours on eight 4090 GPUs. To help the inverse renderer generate higher-quality results, we then resume training it on a 1024 resolution for 2K iterations with a batch size of 8 for about 2 days on eight 4090 GPUs. For the forward renderer, we use Qwen2.5-VL-7B-Instruct[2] to generate a text prompt for each image first. Similar to the inverse renderer, we only train the DiT module using a learning rate of 1e-5 and batch size of 48 for nearly 40 hours on 7 L40s GPUs. The LoRA [8] finetuning with feature distillation costs 8 hours on four 4090 GPUs, with a learning rate of 1e-4 and the LoRA rank of 32.

D. Dataset construction details

Our WeatherSynthetic contains 35 thousand images of autonomous driving scenes under various weather conditions, simulating diverse weather and lighting scenarios. It includes detailed annotations of intrinsic maps, including albedo, normal, roughness, metallicity, and irradiance. A comparison between our dataset and other datasets is shown in Tab. 1. Additionally, to facilitate generation tasks, we provide a detailed prompt for each image, including scene elements, weather, time of day, and more. More examples are shown in Fig. 8. The WeatherReal contains 18 thousand real-world images. We first leverage Qwen2.5-VL-7B-Instruct to filter sunny scenes and then use the inverse rendering model to obtain intrinsic maps. Specifically, we use "Generate a sentence describing this {weather_type} weather driving scene." as the template, guiding Qwen2.5-VL-7B-Instruct to generate a reasonable and appropriately concise description. More examples are shown in Fig. 9.

[2] https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct

Figure 3. Ablation on intrinsic representation. We show a comparison between the model trained on intrinsic representation (Ours) and RGB image (RGB repr.).

E. More ablation results

Effect of intrinsic representation. The intrinsic representation imposes a physical inductive bias that enables high-fidelity editing with limited training data. We replace the intrinsic representation with a clean-weather image similar to WeatherWeaver [18], and train the model with the same setup as the forward renderer. We find that the model struggles to maintain geometry and material details when modifying the weather conditions, as shown in Fig. 3. Moreover, our intrinsic-based editing pipeline supports material editing better.

Effect of IMAA. Following the heuristic-guided progressive training strategy proposed in Sec. C.4, we first train IMAA and the Diffusion model, respectively. The pretrained IMAA can be applied to the finetuned Diffusion model without additional finetuning, as shown in row 2 of Tab. 2, and the performance of the original Diffusion model without further finetuning is reported in row 4 in Tab. 2. IMAA helps the Diffusion model generate more plausible estimations, especially the roughness (PSNR from 22.56 to 23.85) and metallicity (PSNR from 26.30 to 31.15). Then we further finetune the Diffusion model with frozen IMAA, and the results are shown in row 1 of Tab. 2. We also finetune the Diffusion model without IMAA for the same steps, as shown in row 3. After additional finetuning, our full model achieves the best performance on almost all metrics.

We also validate the effect of heuristic-guided progressive training, as shown in row 5 of Tab. 2. The performance of the diffusion model with IMAA training from scratch is similar to the model without IMAA. Moreover, we find that

Table 1. Comparison of our datasets and previous datasets. Our dataset includes both synthetic and real-world data, and simulates a wide variety of weather and lighting conditions. Meanwhile, we provide detailed map annotations for each scene. Each feature is marked as satisfied (✓), partially satisfied (✓), or not satisfied (✗).

	Images	Scene	Source	Weather	Intrinsic map				
					Albedo	Normal	Roughness	Metallicity	Irradiance
InteriorVerse	50K	Indoor	Synthetic	✗	✓	✓	✓	✓	✗
Hypersim	70K	Indoor	Synthetic	✗	✓	✓	✗	✗	✓
Openrooms	118K	Indoor	Synthetic	✗	✓	✓	✓	✗	✗
Matrixcity	316K	City	Synthetic	✓	✓	✓	✓	✓	✗
WeatherSynthetic	35K	City	Synthetic	✓	✓	✓	✓	✓	✓
WeatherReal	18K	City	Real-world	✓	✓	✓	✓	✓	✓

Table 2. Ablation study of IMAA. All results are evaluated on the WeatherSynthetic test set. We highlight the best results in red and the second-best results in orange.

Method	Albedo			Normal			Roughness		Metallic		Irradiance	
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	MAE↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓
Ours (M+, F+)	27.99	0.86	0.35	25.06	0.84	4.24	25.81	0.23	29.29	0.04	29.66	0.22
Ours (M+, F-)	27.49	0.79	0.38	23.85	0.75	4.58	24.53	0.21	31.15	0.04	27.98	0.36
Ours (M-, F+)	26.78	0.84	0.43	23.63	0.79	6.33	24.60	0.25	28.16	0.05	26.99	0.32
Ours (M-, F-)	26.66	0.83	0.45	22.56	0.77	5.15	22.91	0.29	26.30	0.06	27.03	0.34
Ours (M+, no prog)	24.63	0.74	0.47	22.89	0.76	6.35	21.60	0.25	22.37	0.08	26.32	0.45

Note: M+ = with IMAA; M- = without IMAA; F+ = with additional finetune; F- = without additional finetune; no prog = without heuristic-guided progressive training.

the map-aware mask generated by IMAA is close to a zero matrix. These results suggest that without heuristic-guided progressive training, IMAA collapses during training, producing near-zero masks and thus failing to provide effective supervised signals.

Effect of WeatherSynthetic. We train our IntrinsicWeather with only the indoor dataset [24, 32]. We show results in Fig. 4. The estimation is flawed despite the generalization of the diffusion model.

Effect of WeatherReal. We show results of the model trained only on synthetic data in Fig. 5. Due to the domain gap, the performance of the model without WeatherReal in some severe weather is heavily affected.

Effect of distillation for fine-grained weather control We train a LoRA [8] based on the forward renderer trained on the synthetic data without feature distillation. The comparison is shown in 6. Without feature abolition (Ours w/o dis.), the intermediate weather states tend to resemble the two endpoints, lacking a natural transition. When applying distillation (Ours w/ dis.), the results are more natural.

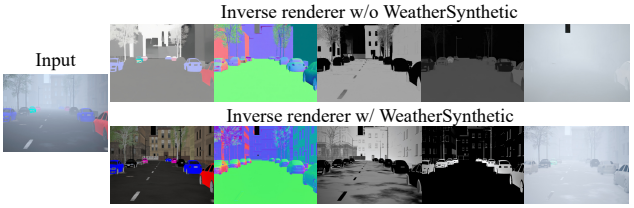

Figure 4. Ablation on WeatherSynthetic. We show a comparison between the model trained only on the indoor dataset and the model trained on the indoor dataset and WeatherSynthetic.

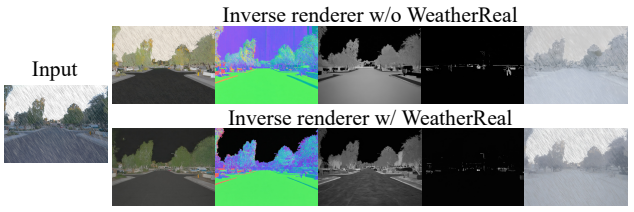

Figure 5. Ablation on WeatherReal. We show a comparison between the model trained with real-world data and the one trained without it.

Figure 6. Ablation on feature distillation. When applying distillation, the intermediate results are more natural.

F. Comparison with image editing methods

F.1. More qualitative comparisons

In Fig. 10, we compare our method's weather removal performance with state-of-the-art pixel-space editing methods (*i.e.*, Flux-Kontext [13] and Qwen-Image-Edit [28]), a classic instruction-based method (Instruct-Pix2Pix [2]), an inversion-based method (*i.e.*, TurboEdit [29]), and weather specific editing method (*i.e.*, WeatherWeaver [18]). Our method removes snowflakes, raindrops, and snow cover while maintaining material and geometric details. Flux-Kontext struggles to remove weather artifacts across most scenes effectively. Qwen-Image-Edit partially removes the air particles, but fails to remove the splash and rain trace. Instruct-Pix2Pix tends to misinterpret the task, often producing stylized outputs rather than realistic, weather-free reconstructions. WeatherWeaver can remove partial weather effects, but fails to keep original details, and generates blurry images. Inversion-based methods struggle to handle global editing tasks such as weather editing, and they often produce irrelevant or unintended modifications. Moreover, other pixel-space editing methods struggle to preserve material and geometry details when synthesizing new weather conditions.

In Fig. 11 we show the performance of weather synthesis. Our method leverages inverse rendering to decouple the material and lighting first, and then re-renders the image under new weather and illumination conditions. This framework helps modify the lighting more naturally. Flux-Kontext and Qwen-image-Edit tend to generate unnatural phenomena, such as significant shadows on an overcast day.

Table 3. User preference across different weather conditions.

Method	User Preference (%)↑	
	Weather Removal	Weather Synthesis
Ours	77.07	86.28
Flux-Kontext	13.70	7.58
Qwen-Image-Edit	7.90	1.32
WeatherWeaver	1.33	4.82

F.2. More results on real-world dataset

In Fig. 12, we present the full weather editing process of our IntrinsicWeather. Note the consistent preservation of scene geometry and object identity across all generated weather conditions, demonstrating the robustness of our editing framework in intrinsic space.

In Fig. 13, we present weather editing on the ACDC dataset [25]. The weather-aware inverse renderer decouples different weather conditions from material and geometry. Then the weather-conditioned forward renderer re-renders images under target weather conditions. We show robust editing under different original weather conditions (rows 1-2: fog, rows 3-4: snow, rows 5-6: rain).

F.3. User study

We conducted a user study to evaluate the consistency and realism of different editing methods. Eight representative cases were prepared, each showing our results alongside several baselines under various weather conditions. For each task, participants saw randomized, anonymized outputs from all methods and selected the most realistic. The order was randomized per participant. A total of 61 participants were asked to vote for the most realistic and consistent result in each case. As shown in Tab. 3, our methods won the most preference in both weather removal and synthesis tasks.

G. Validation on ACDC dataset

Setup. For the ACDC [25] benchmark, we apply our IntrinsicWeather to every image in the validation set and obtain a re-rendered version under the target weather condition. We then feed both the original images and the edited images into off-the-shelf DETR[3] and Segformer models[4] (without any further fine-tuning) and compare their detection/segmentation accuracy. This allows us to quantify the contribution of our editing model to downstream perception robustness.

[3] https://huggingface.co/facebook/detr-resnet-50

[4] https://huggingface.co/nvidia/segformer-b5-finetuned-cityscapes-1024-1024

Table 4. Quantitative evaluations of our method against existing methods on indoor scenes. Albedo, normal, roughness, and metallicity evaluations are conducted on the InteriorVerse dataset, and irradiance evaluation is conducted on the Hypersim dataset. Due to corruption in the metallicity channel of the open-source RGB↔X model, we exclude it from the comparison. We highlight the best results in red and the second-best results in orange.

Method	Albedo			Normal			Roughness		Metallic		Irradiance	
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	MAE↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓
IID	17.40	0.80	0.22	–	–	–	14.58	0.27	13.83	0.28	–	–
RGB↔X	16.40	0.78	0.54	17.18	0.78	5.53	10.90	0.53	–	–	14.10	0.22
GeoWizard	–	–	–	10.40	0.62	6.14	–	–	–	–	–	–
IDArb	6.04	0.48	0.65	11.70	0.66	10.32	9.60	0.59	9.09	0.62	–	–
DiffusionRenderer	22.40	0.87	0.19	20.97	0.83	10.90	10.63	0.57	12.08	0.36	–	–
Ours	23.58	0.93	0.18	22.88	0.85	3.11	17.90	0.18	20.95	0.18	16.54	0.20
Ours (w/o IMAA)	22.19	0.88	0.26	20.01	0.83	4.58	15.57	0.20	18.16	0.22	15.08	0.33

H. More results of inverse rendering

H.1. Indoor scenes

We show quantitative comparison for indoor scenes in Tab. 4, and our IntrinsicWeather outperforms state-of-the-art methods. Qualitative comparison is shown in Fig. 14 and more qualitative results in Fig. 16. Our IntrinsicWeather set a special weather controller to present "indoor". Our method provides an accurate estimation of all intrinsic maps. As shown in Fig. 14, the albedo maps predicted by other methods exhibit color deviations on the ceiling, whereas our method correctly produces a prediction that is close to white. Our method generates constant and sharp normal predictions, while others fail to recover geometry details. In the roughness and metallicity map, our IntrinsicWeather can predict maps with sharp boundaries, while other methods, such as IID [12], RGB↔X [31], and DiffusionRenderer [17] tend to produce predictions with blurred edges. Last, our IntrinsicWeather and RGB↔X both generate accurate irradiance maps.

H.2. Autonomous driving scenes

Synthetic scenes. An overall qualitative comparison is shown in Fig. 7. On a sunny day (row 1), both ours and DiffusionRenderer produce reasonable estimations, whereas the other methods fail. Under adverse weather conditions (rows 2–5), the performance of all other methods deteriorates significantly due to the presence of airborne particles. In contrast, our method effectively removes the influence of fog, snowflakes, and rain, yielding clean and faithful predictions. We show extra qualitative comparison between our IntrinsicWeather and other methods in Figs. 17 to 19. More results are shown in Fig. 20. After fine-tuning, IID and RGB↔X can generate reasonable results on autonomous driving scenes. But in detail, these methods fail to generate an accurate estimation (*e.g.*, the albedo of the car hiding in the shadow). We further validate the consistency of our inverse renderer across weather conditions (see Fig. 21). For each scene, we run the inverse renderer on images captured under different weather types and compute the PSNR between the recovered intrinsic maps and those obtained under sunny conditions.

Real-world scenes. We show a qualitative comparison in Fig. 22. Our method remains robust under various weather conditions. We show more results on real-world scenes in Fig. 23. The figure includes heavy rain (rows 3, 4), blizzard (rows 2, 6, and 7), dense fog (row 8), and sandstorm scenarios (row 5). In particular, we additionally showcase the strong reflections caused by wet roads after rain (row 1) and the impact of accumulated snow (rows 6, 7) on model stability. As can be seen, our model produces reasonable predictions under all these challenging conditions.

H.3. Other outdoor scenes.

In this subsection, we will show results on several out-of-distribution outdoor scenes that never appear in our training dataset. Surprisingly, due to the generalization ability of the diffusion model, our IntrinsicWeather can generate decent results.

I. Failure case

We show some typical failure cases in Fig. 26. When facing extreme heavy weather, such as dense fog, our inverse renderer may generate hallucinations in the background where information is completely occluded. Though the forward renderer can remain robust when the intrinsic maps are imprecise, the details in the re-rendered background are consequently unreliable and may not reflect the true scene.

References

[1] Sai Bi, Xiaoguang Han, and Yizhou Yu. An l 1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions On Graphics (TOG)*, 34 (4):1–12, 2015. 1

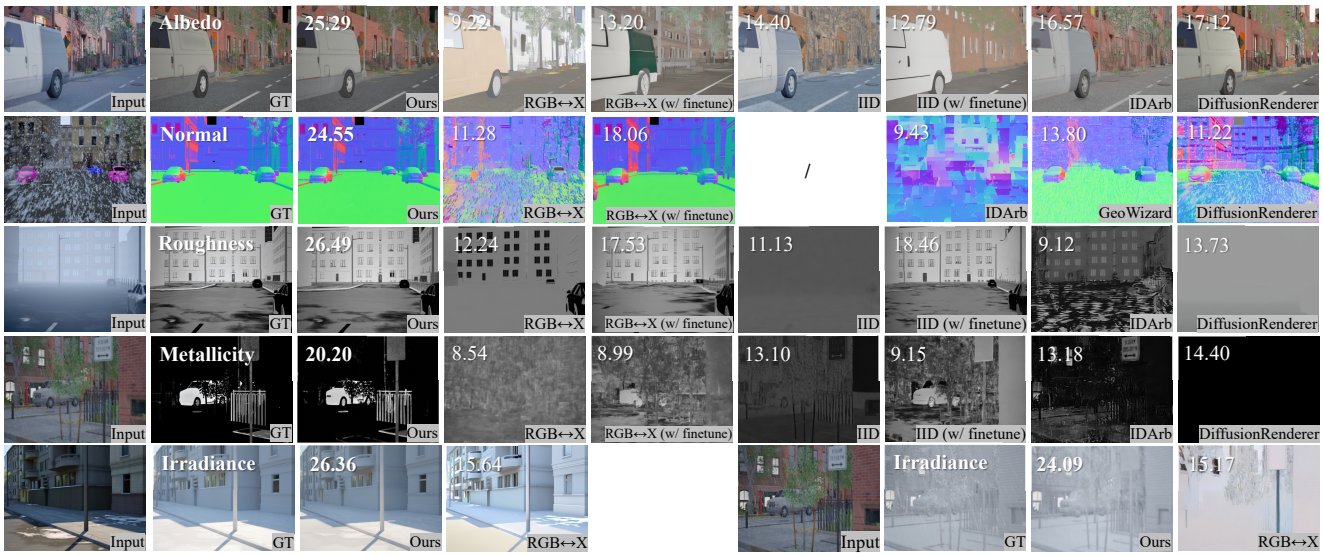

Figure 7. Comparisons of inverse rendering results under different weather conditions with PSNR values of each result. The highest PSNR is marked in bold. We show the comparison of all the maps in the supplementary material.

[2] 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. 1, 5

[3] Graham D Finlayson, Mark S Drew, and Cheng Lu. Intrinsic images by entropy minimization. In *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III 8*, pages 582–595. Springer, 2004. 1

[4] Xiao Fu, Wei Yin, Mu Hu, Kaixuan Wang, Yuexin Ma, Ping Tan, Shaojie Shen, Dahua Lin, and Xiaoxiao Long. Geowizard: Unleashing the diffusion priors for 3d geometry estimation from a single image. In *European Conference on Computer Vision*, pages 241–258. Springer, 2024. 14

[5] Elena Garces, Adolfo Munoz, Jorge Lopez-Moreno, and Diego Gutierrez. Intrinsic images by clustering. In *Computer graphics forum*, pages 1415–1424. Wiley Online Library, 2012. 1

[6] Martin Hahner, Dengxin Dai, Christos Sakaridis, Jan-Nico Zaech, and Luc Van Gool. Semantic understanding of foggy scenes with purely synthetic data. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3675–3681, 2019. 21

[7] Berthold KP Horn. Determining lightness from an image. *Computer graphics and image processing*, 3(4):277–299, 1974. 1

[8] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 3, 4

[9] Michael Janner, Jiajun Wu, Tejas D Kulkarni, Ilker Yildirim, and Josh Tenenbaum. Self-supervised intrinsic image decomposition. *Advances in neural information processing systems*, 30, 2017. 1

[10] Mourad A Kenk and Mahmoud Hassaballah. Dawn: vehicle detection in adverse weather nature dataset. *arXiv preprint arXiv:2008.05402*, 2020. 21

[11] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013. 1

[12] Peter Kocsis, Vincent Sitzmann, and Matthias Nießner. Intrinsic image diffusion for indoor single-view material estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5198–5208, 2024. 6, 14

[13] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. 5

[14] Edwin H Land and John J McCann. Lightness and retinex theory. *Journal of the Optical society of America*, 61(1):1–11, 1971. 1

[15] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2475–2484, 2020. 1

[16] Zhibing Li, Tong Wu, Jing Tan, Mengchen Zhang, Jiaqi Wang, and Dahua Lin. Idarb: Intrinsic decomposition for arbitrary number of input views and illuminations. *arXiv preprint arXiv:2412.12083*, 2024. 14

[17] Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon

Hasselgren, Zhi-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, et al. Diffusionrenderer: Neural inverse and forward rendering with video diffusion models. *arXiv preprint arXiv:2501.18590*, 2025. 6, 14

[18] Chih-Hao Lin, Zian Wang, Ruofan Liang, Yuxuan Zhang, Sanja Fidler, Shenlong Wang, and Zan Gojcic. Controllable weather synthesis and removal with video diffusion models. *arXiv preprint arXiv:2505.00704*, 2025. 3, 5

[19] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 1

[20] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9970–9980, 2024. 1

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3

[22] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 1

[23] Julien Philip, Sébastien Morgenthaler, Michaël Gharbi, and George Drettakis. Free-viewpoint indoor neural relighting from multi-view stereo, 2021. 1

[24] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10912–10922, 2021. 1, 3, 4

[25] Christos Sakaridis, Haoran Wang, Ke Li, René Zurbrügg, Arpit Jadon, Wim Abbeloos, Daniel Olmeda Reino, Luc Van Gool, and Dengxin Dai. Acdc: The adverse conditions dataset with correspondences for robust semantic driving scene perception. *arXiv preprint arXiv:2104.13395*, 2021. 5

[26] Jeya Maria Jose Valanarasu, Rajeev Yasarla, and Vishal M. Patel. Transweather: Transformer-based restoration of images degraded by adverse weather conditions, 2021. 21

[27] Zian Wang, Jonah Philion, Sanja Fidler, and Jan Kautz. Learning indoor inverse rendering with 3d spatially-varying lighting, 2021. 1

[28] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. 5

[29] Zongze Wu, Nicholas Kolkin, Jonathan Brandt, Richard Zhang, and Eli Shechtman. Turboedit: Instant text-based image editing, 2024. 5

[30] Ye Yu and William AP Smith. Inverserendernet: Learning single image inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3155–3164, 2019. 1

[31] Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. Rgb↔x: Image decomposition and synthesis using material- and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*, New York, NY, USA, 2024. Association for Computing Machinery. 1, 6, 14

[32] 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. 1, 3, 4, 15

Figure 8. More samples of WeatherSynthetic.

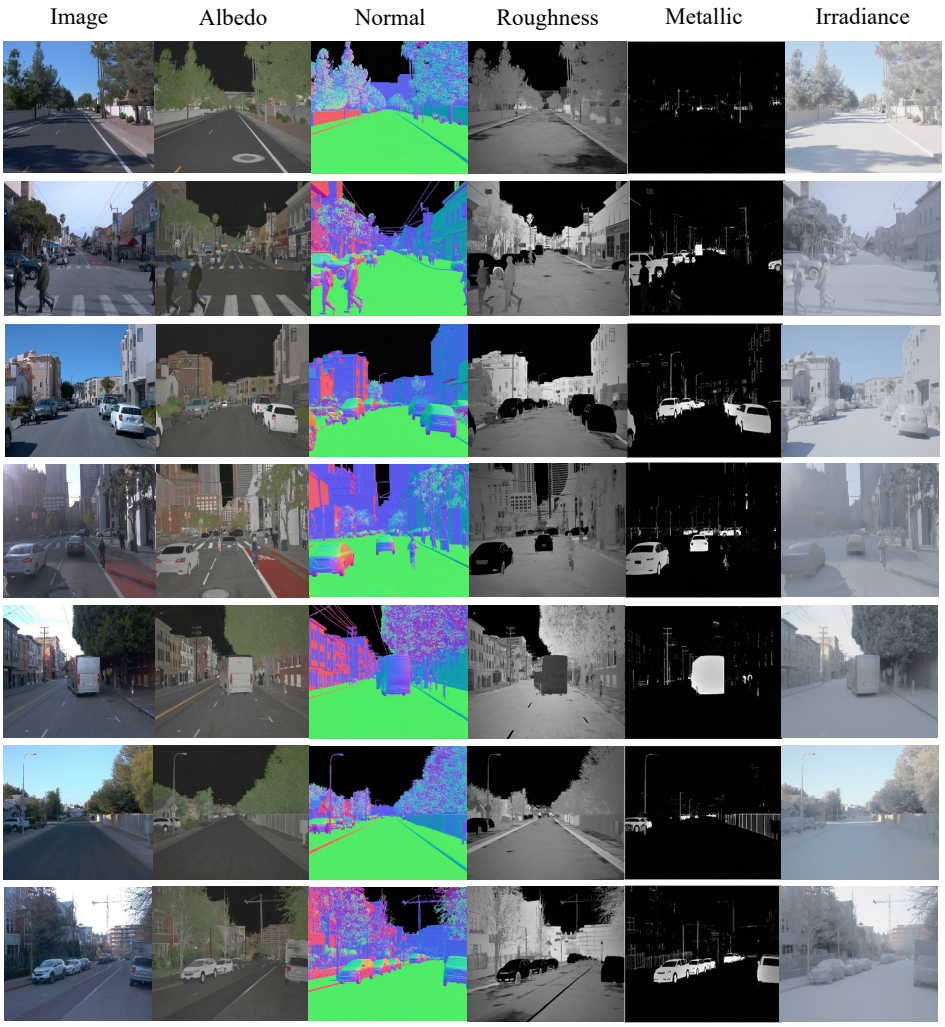

Figure 9. Samples of WeatherReal. We choose the pseudo-labels generated by the inverse renderer randomly and check the quality. Most pseudo labels are high-quality.

Figure 10. Comparison of weather removal with image editing methods.

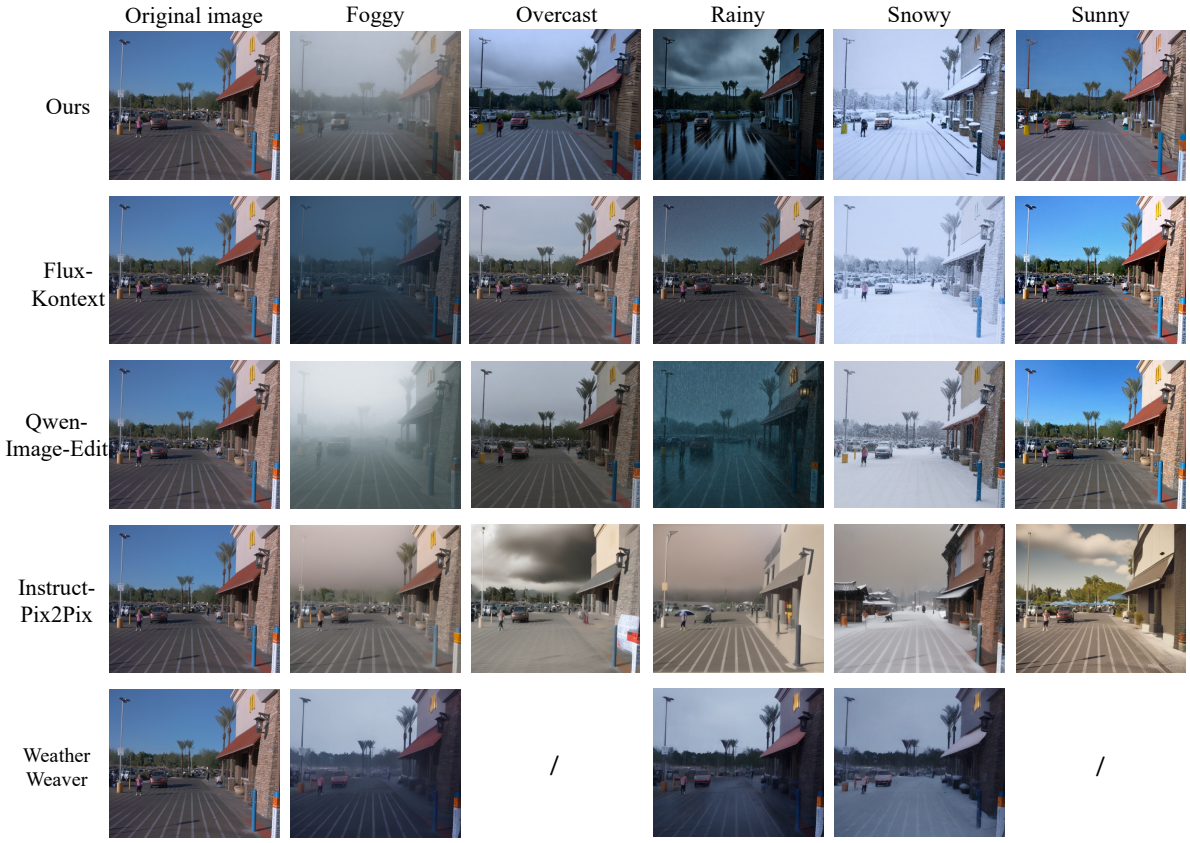

Figure 11. Comparison of weather synthesis with image editing methods.

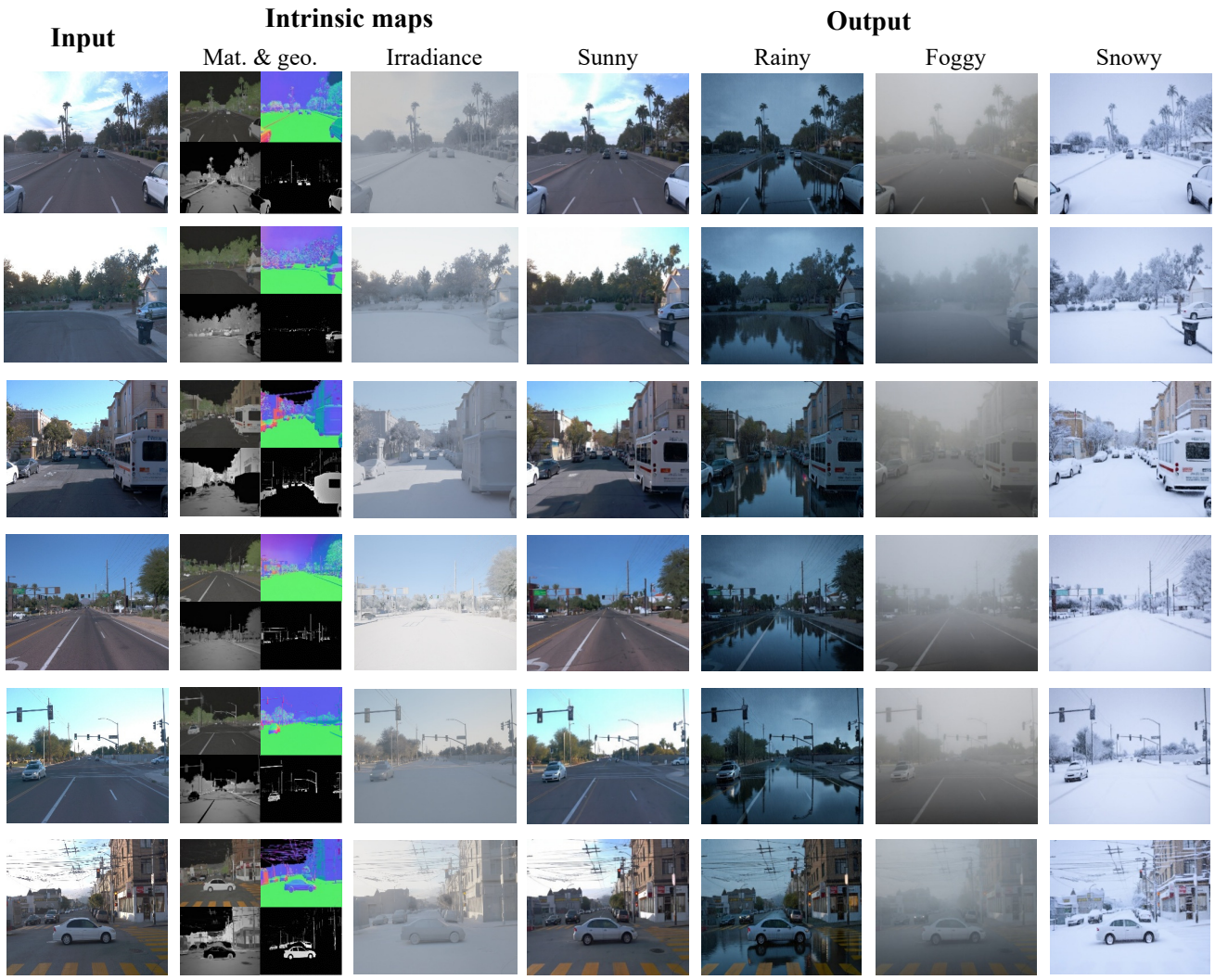

Figure 12. Examples of weather editing on the Waymo dataset. We first leverage the inverse render to obtain the weather-invariant material and geometry maps, along with an irradiance map to capture lighting and weather effects. For the sunny condition, we use all the maps, including the irradiance map, which forces the re-rendered image to have the same illumination as the input. For the other weather conditions, we use the material and geometry maps as input to the forward renderer. Across diverse driving scenes, our model consistently produces coherent intrinsic decomposition and weather editing, demonstrating strong domain generalization.

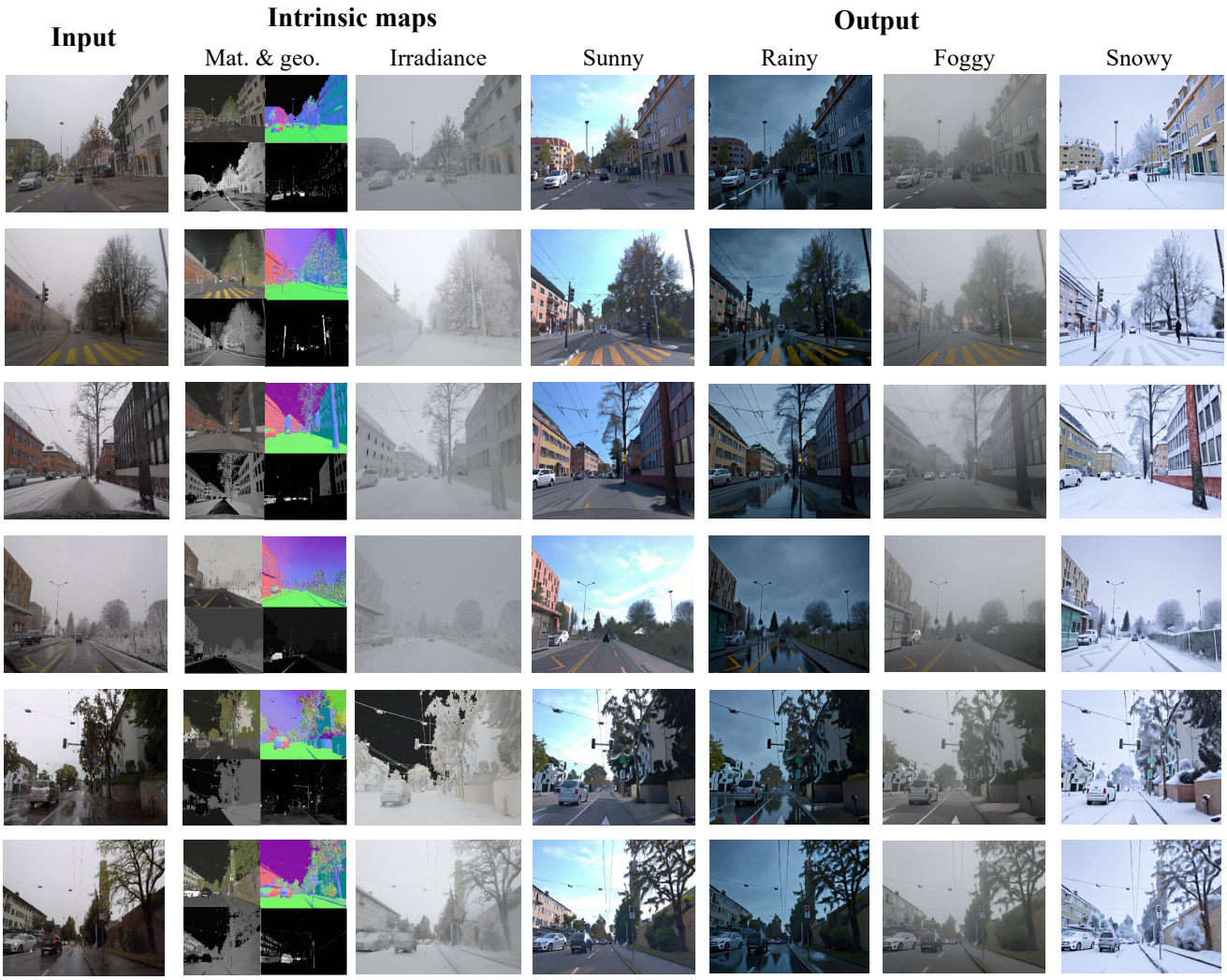

Figure 13. Examples of weather editing on the ACDC dataset. Our weather-aware inverse renderer decouples weather effects from material and geometry, helping clean and consistent weather editing. Across diverse weather conditions, our model consistently produces coherent intrinsic decomposition and weather editing, demonstrating strong domain generalization.

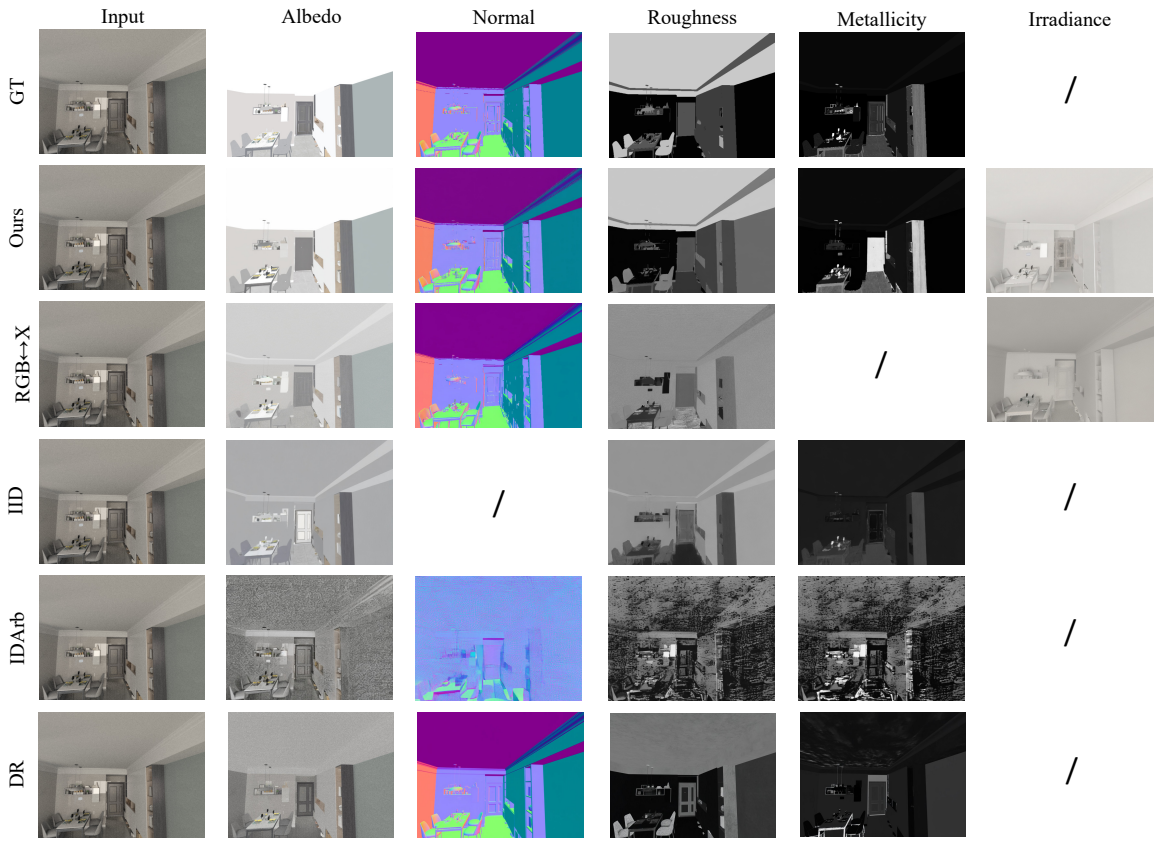

Figure 14. Comparison between IntrinsicWeather and existing methods on indoor scenes. Our method provides more accurate estimations compared with RGB↔X [31], IID [12], GeoWizard [4], IDArb [16], and DiffusionRenderer (DR) [17].

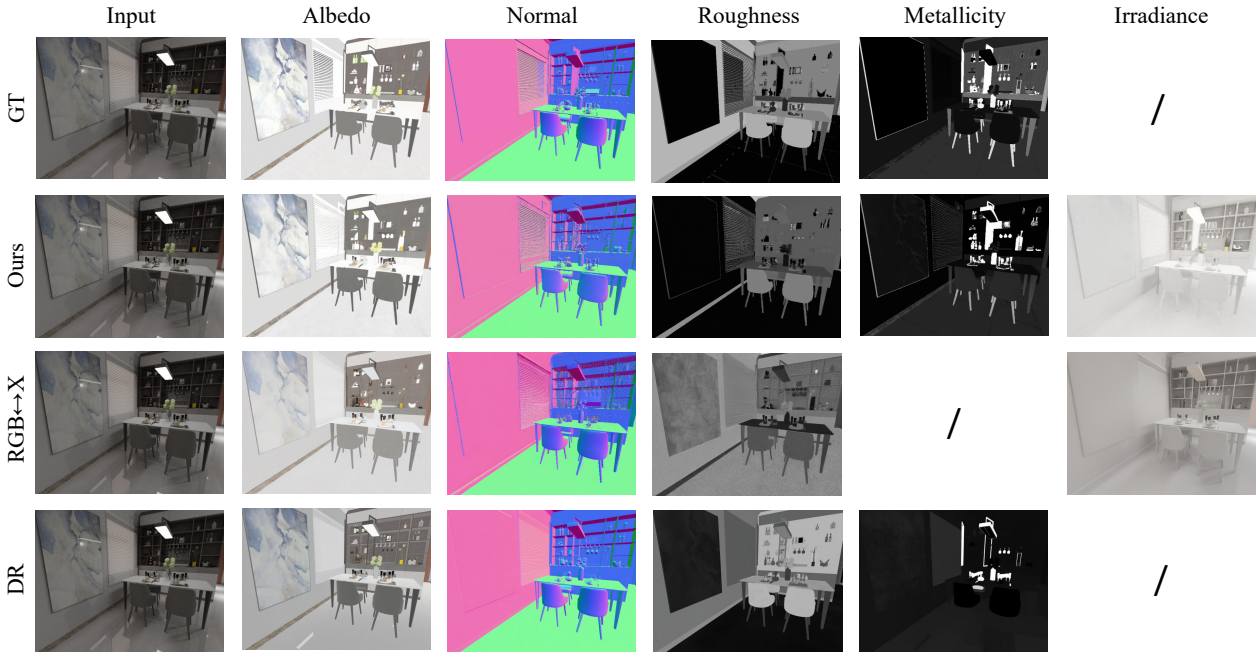

Figure 15. Extra comparison between IntrinsicWeather and existing methods on indoor scenes.

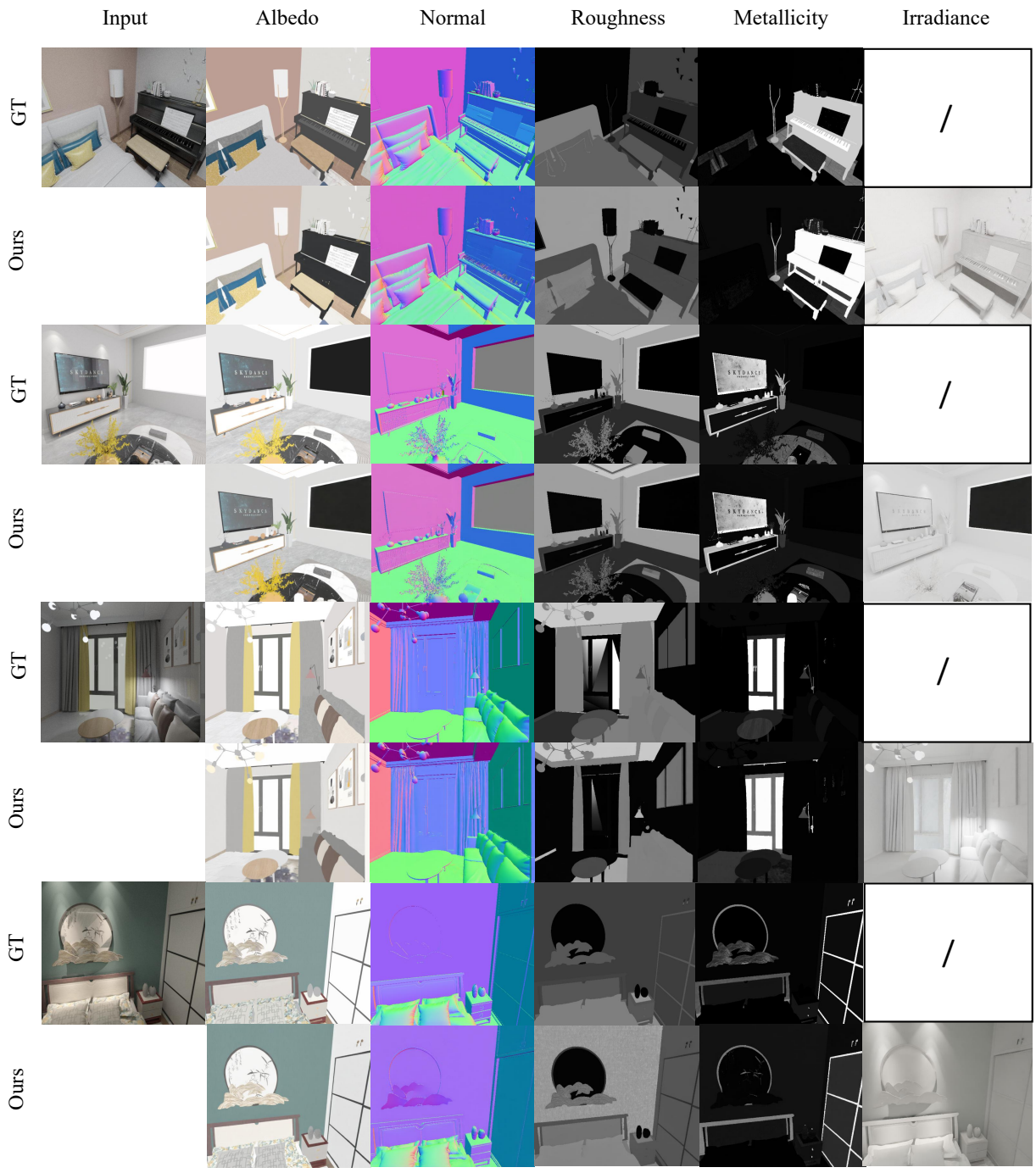

Figure 16. More results on indoor scenes. The inputs are randomly sampled from the test set of InteriorVerse [32]. The irradiance map has no GT, but our IntrinsicWeather can provide accurate estimations.

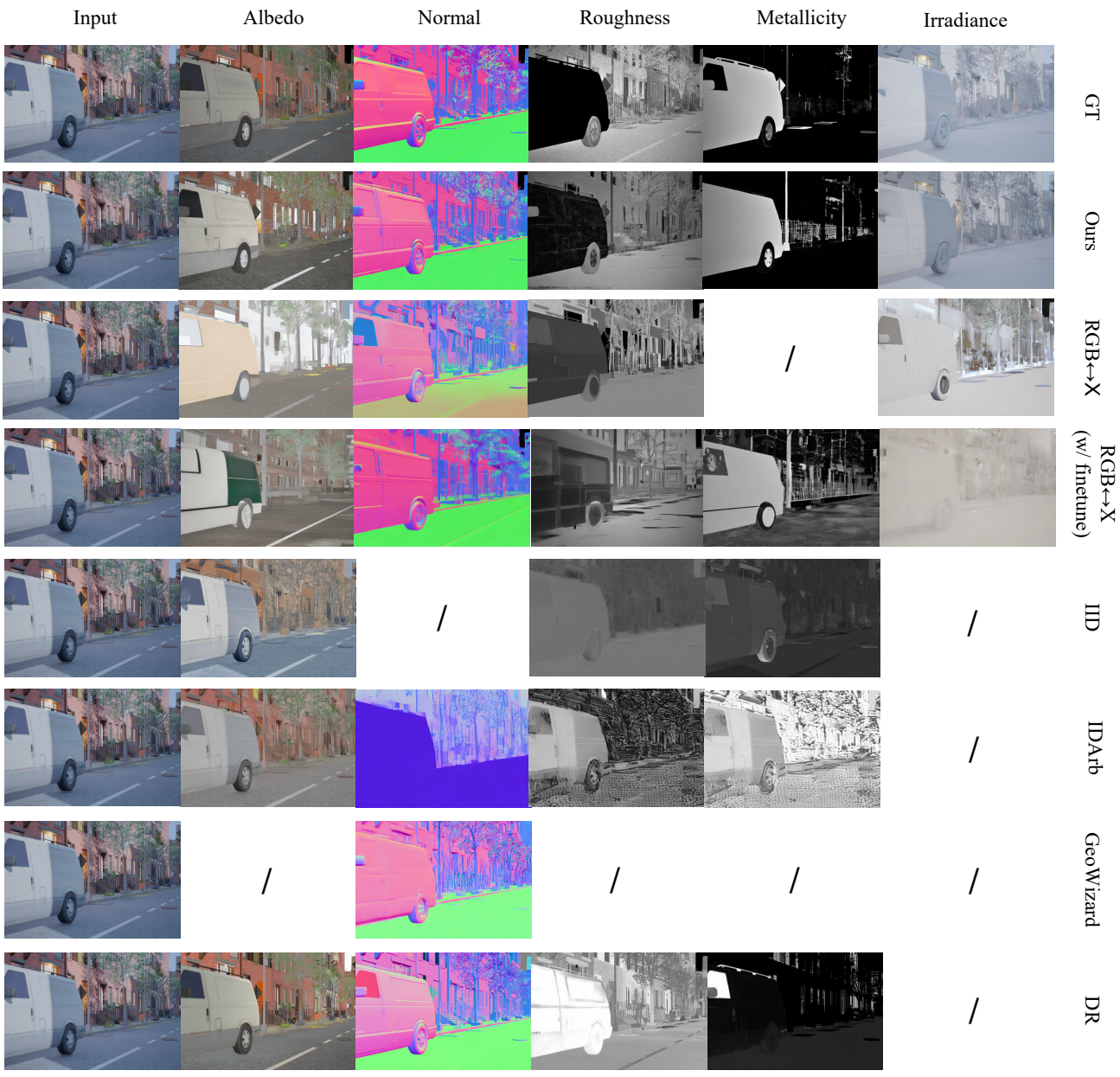

Figure 17. Full comparison between IntrinsicWeather and other methods.

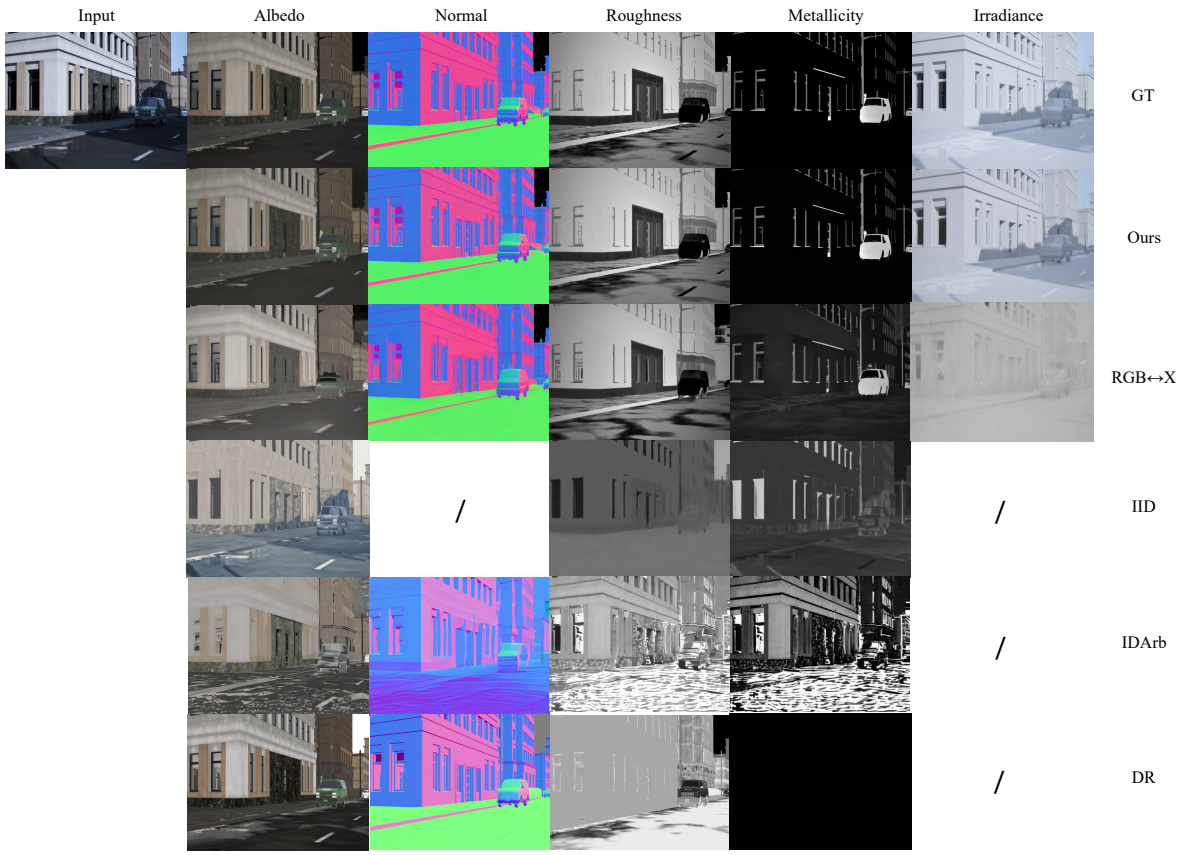

Figure 18. Qualitative comparison between IntrinsicWeather and other methods.

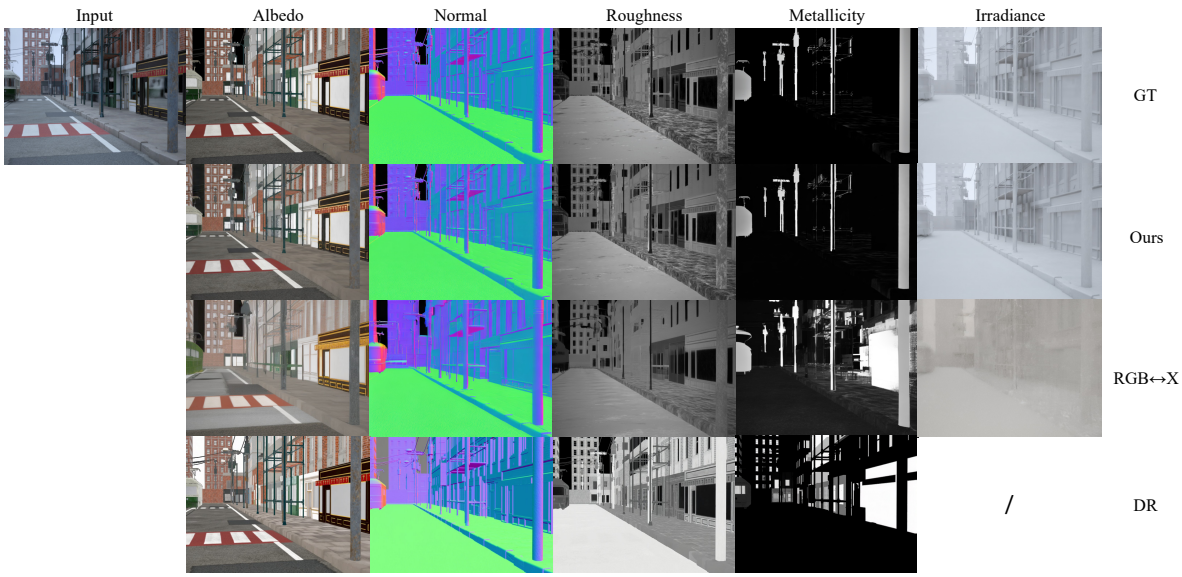

Figure 19. Extra qualitative comparison.

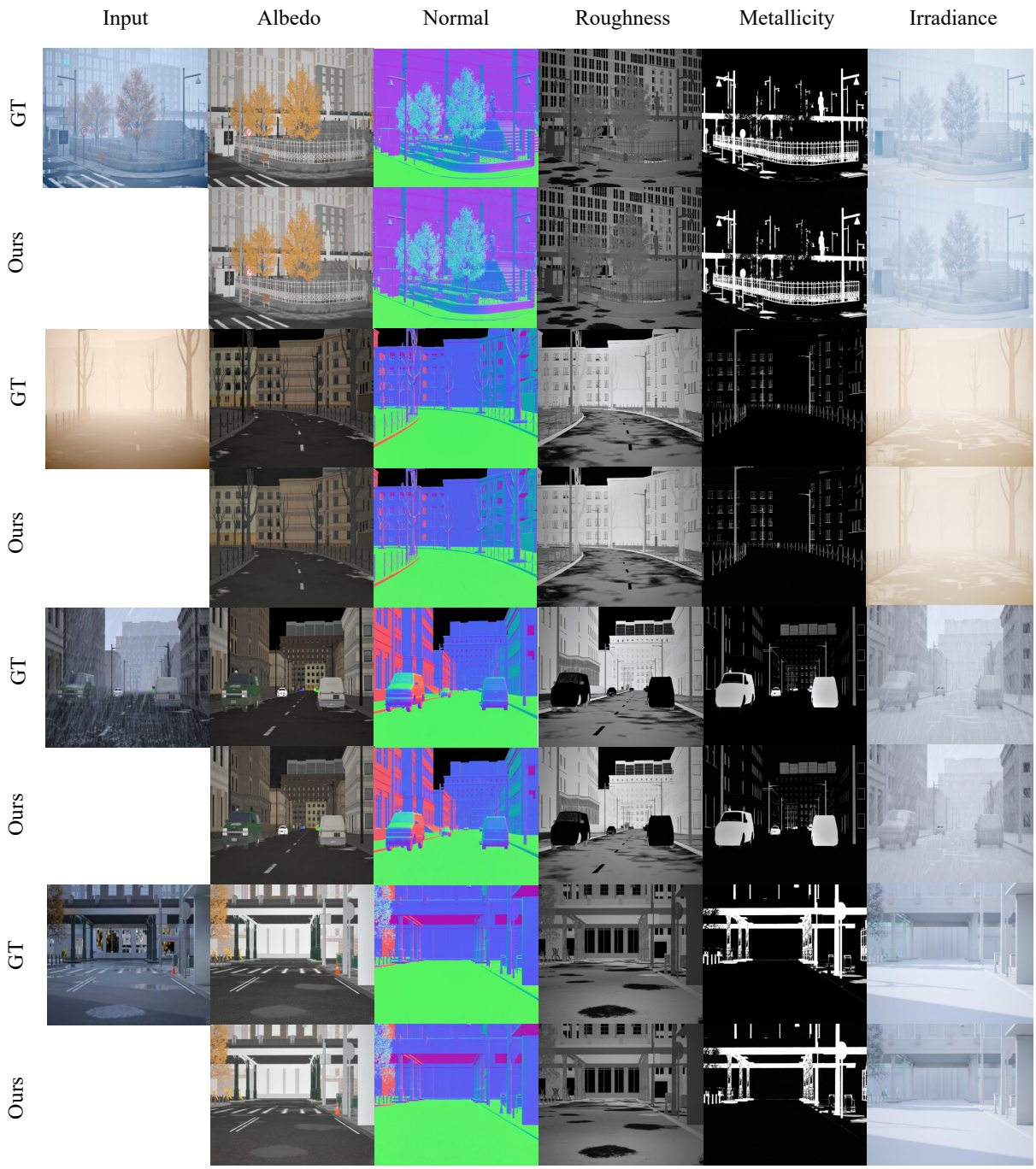

Figure 20. More results on synthetic driving scenes.

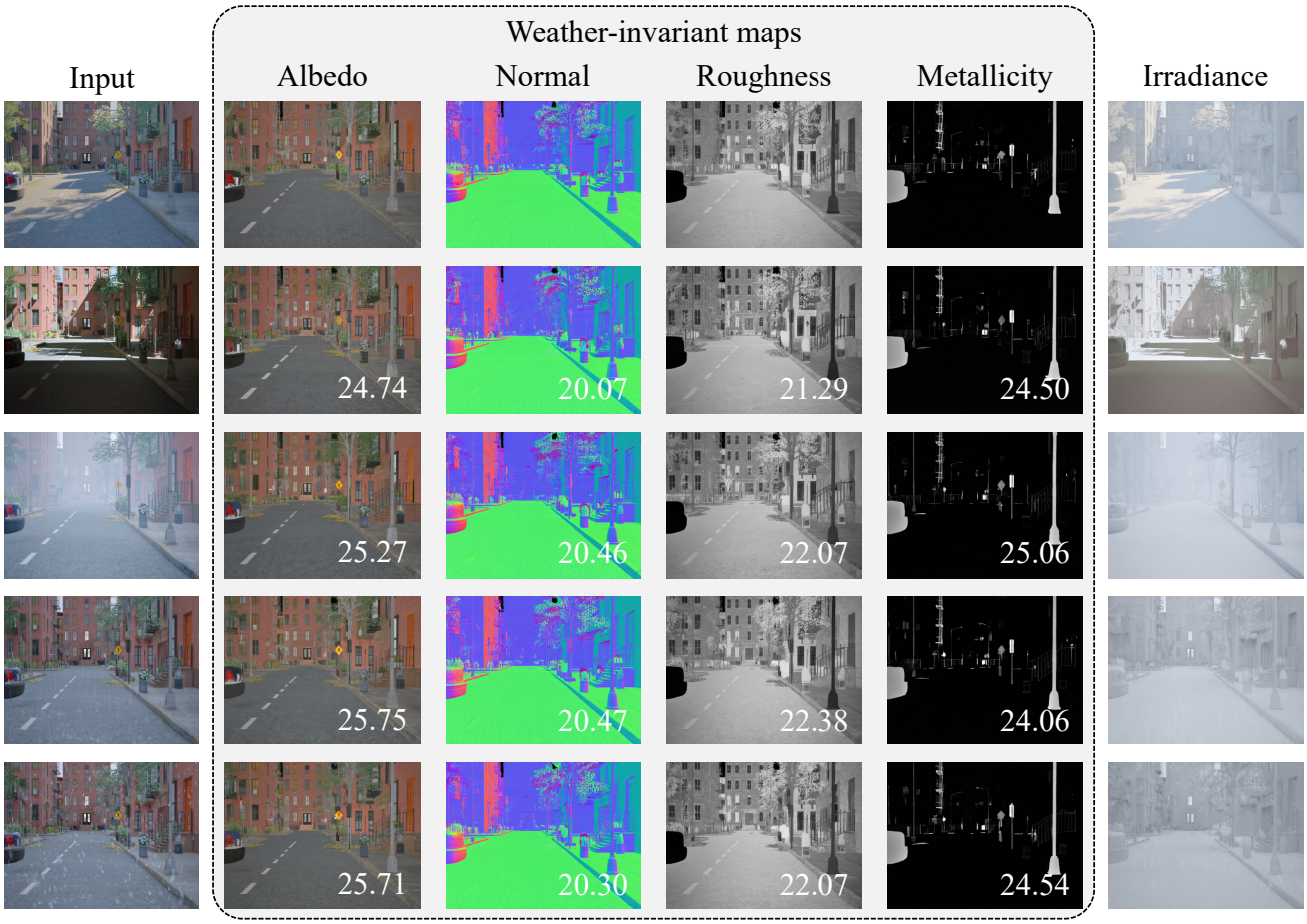

Figure 21. Validation of the consistency of the inverse renderer across weather conditions. PSNR between the recovered intrinsic maps and those obtained under sunny conditions is reported.

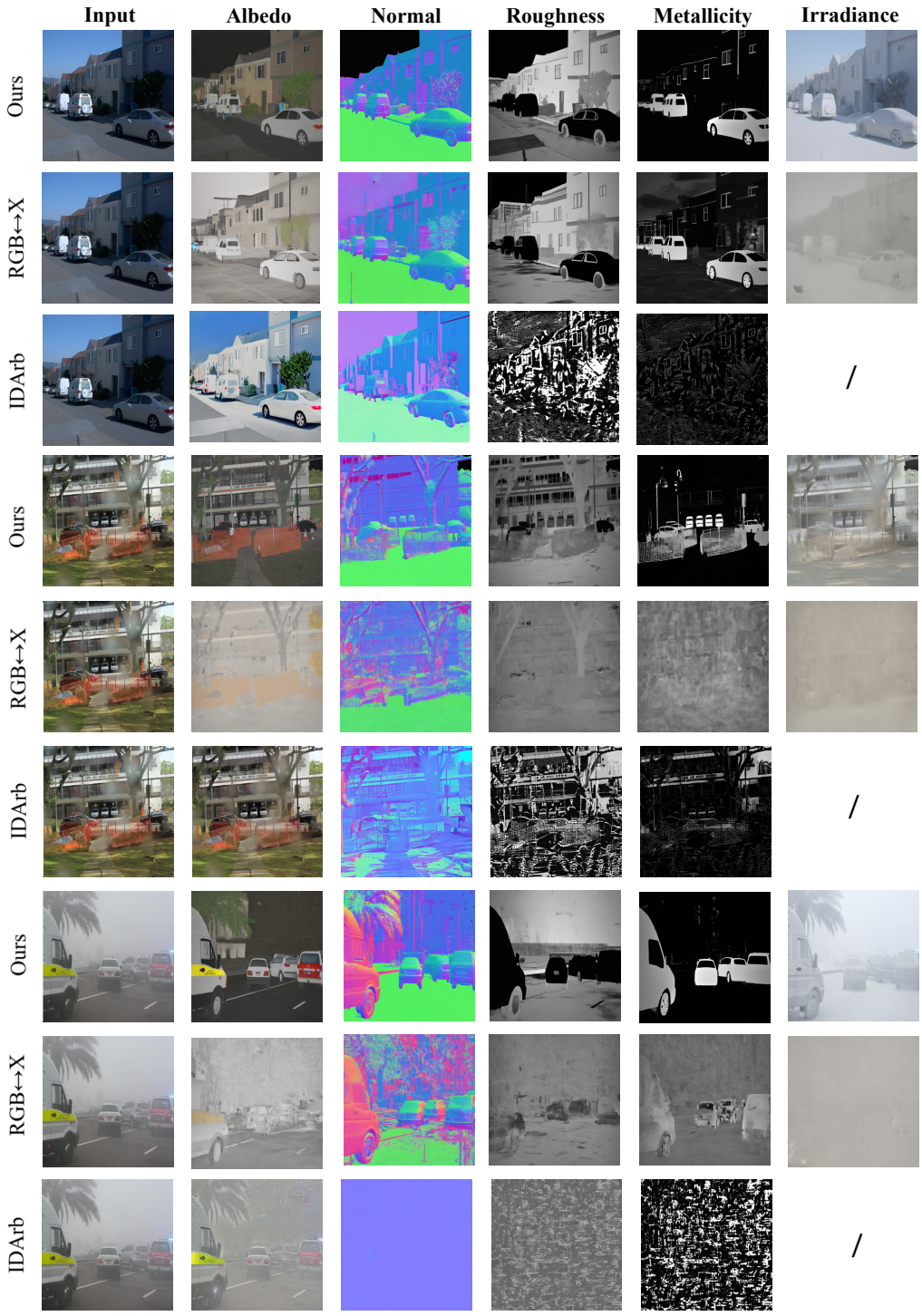

Figure 22. Qualitative comparison on real-world data. Our IntrinsicWeather estimates reasonable results under various weather conditions, while other methods are affected by weather.

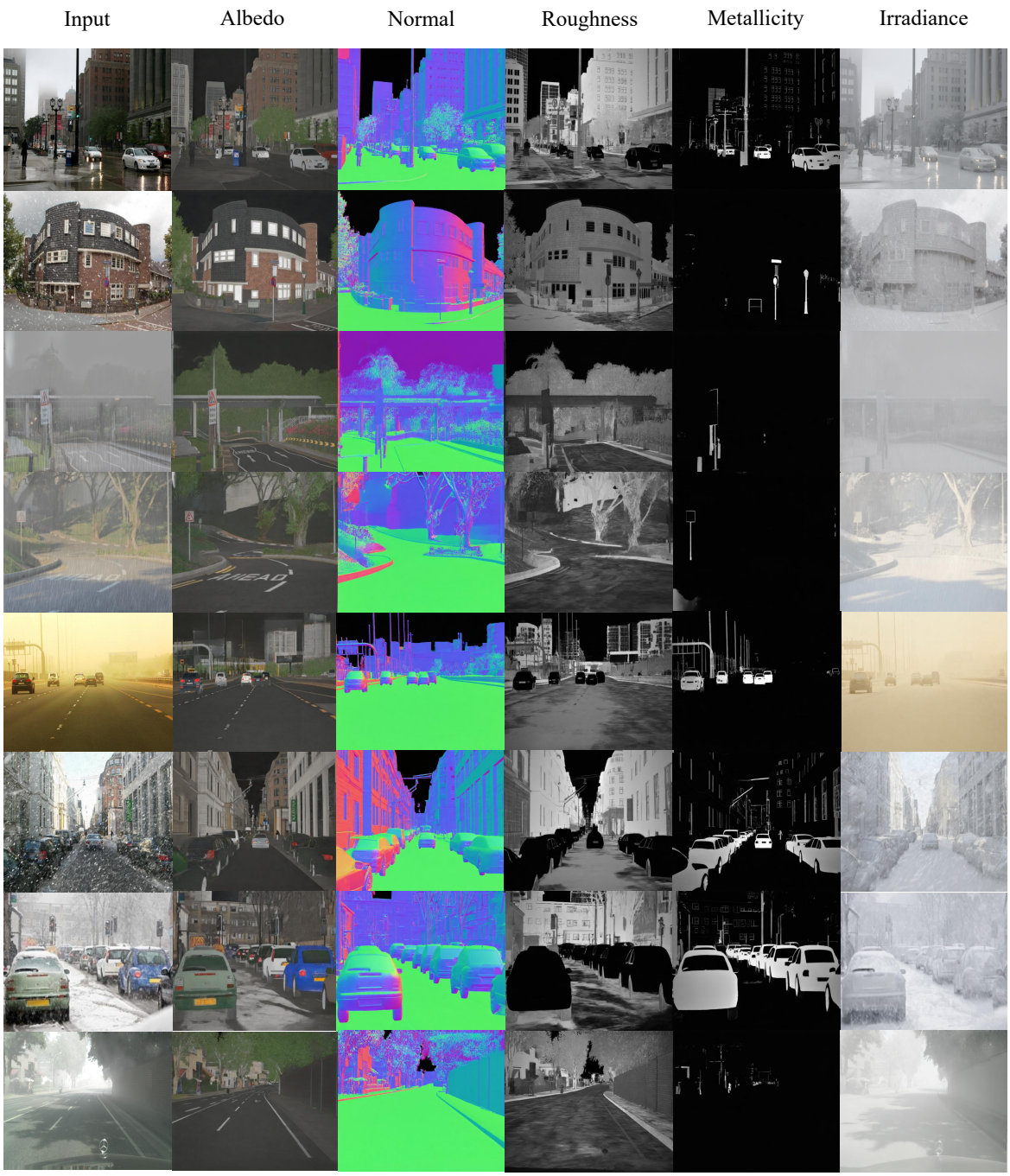

Figure 23. More results on real-world data with diverse weather and lighting conditions. The images are randomly sampled from TranWeather [26], FogCityScapes [6] and DAWN [10].

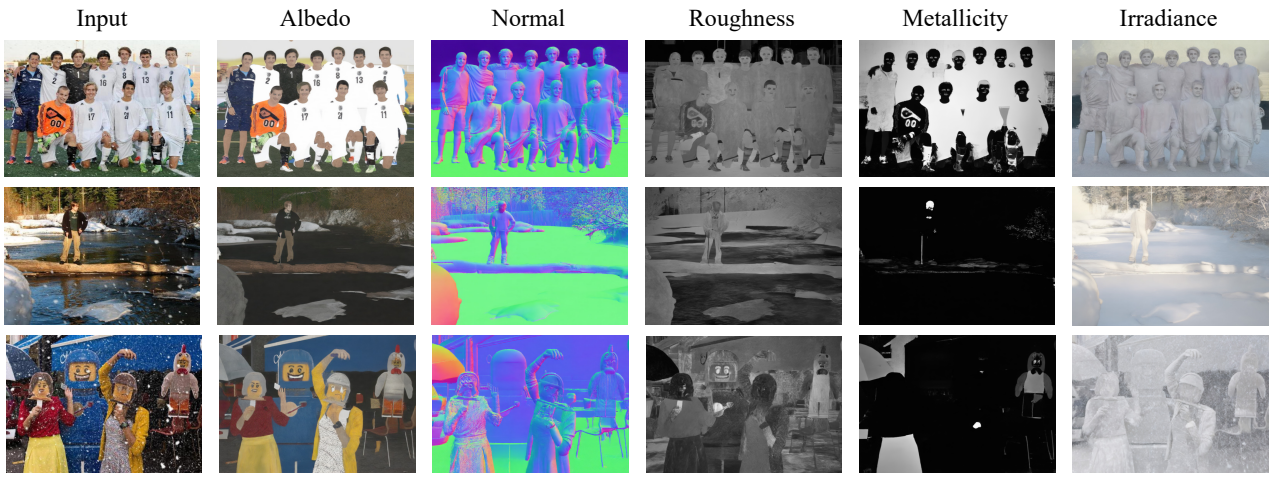

Figure 24. Examples of out-of-distribution data. Humans never appear in our training dataset, but our IntrinsicWeather can generate reasonable results while deriving snowflakes in the air.

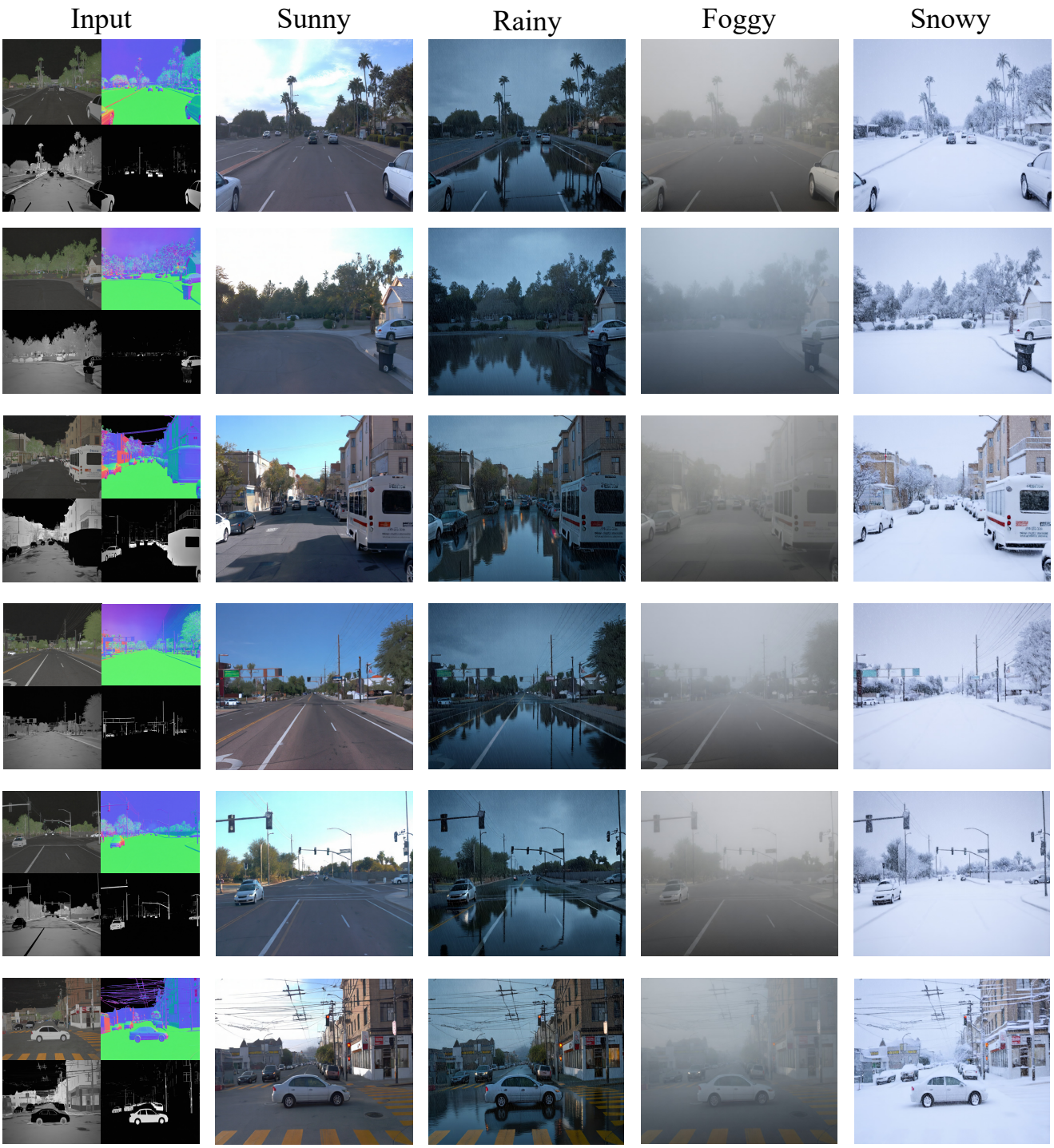

Figure 25. Examples of forward rendering on the Waymo dataset.

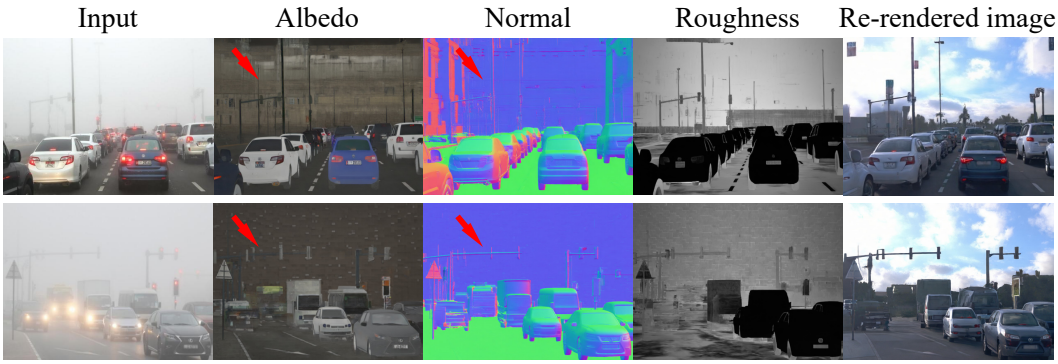

Figure 26. Failure case.