A. Implementation Details

A.1. Inpainting algorithm

The pseudocode of the iterative inpainting algorithm described in Section 3.2 is given in Algorithm 1. Our implementation uses $\gamma = 0.8$, $k = 2$, and $N = 30$. The algorithm repeatedly invokes the INPAINT function, which stands for an inpainting algorithm based on SDEdit [41] as implemented in the Diffusers library [55]. For completeness, we include its pseudocode in Algorithm 0. This algorithm requires no modification to the diffusion model and resembles standard diffusion sampling except the "imputing" step in Line 16.

Algorithm 0: Inpainting using Diffusion Model

1: **function** UPDATE($\mathbf{z}, t, \epsilon$)
2: **return** $\sqrt{\alpha_{t-1}}\left(\frac{\mathbf{z}-\sqrt{1-\alpha_t}\epsilon}{\sqrt{\alpha}}\right) + \sqrt{1-\alpha_{t-1}}\epsilon$
3: **end function**
4:
5: // **input:** latent code of input image $\mathbf{z}_I$,
6: // initial latent code $\mathbf{z}$, // inpainting mask M
7: // conditioning signal (e.g. text, depth map) $\mathbf{C}$
8: // timestep to start denoising (denoising-start).
9: // **output:** Input image with an inpainted chrome ball.
10: **function** INPAINT($\mathbf{z}_I, \mathbf{z}, \mathbf{M}, \mathbf{C}$, denoising-start $= T$)
11: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
12: **for** $i \in \{$denoising-start$, \ldots, 1\}$ **do**
13: $\epsilon \leftarrow \epsilon_\theta(\mathbf{z}, t, \mathbf{C})$
14: $\mathbf{z} \leftarrow$ UPDATE$(\mathbf{z}, t, \epsilon)$
15: $\mathbf{z}'_I \leftarrow \sqrt{\alpha_t}\mathbf{z}_I + \sqrt{1-\alpha_t}\epsilon$
16: $\mathbf{z} \leftarrow (1-\mathbf{M}) \odot \mathbf{z}'_I + \mathbf{M} \odot \mathbf{z}$
17: **end for**
18: $\epsilon \leftarrow \epsilon_\theta(\mathbf{z}, 0, \mathbf{C})$
19: $\mathbf{z} \leftarrow$ UPDATE$(\mathbf{z}, 0, \epsilon)$
20: **return** DECODE$(\mathbf{z})$
21: **end function**

A.2. HDR merging algorithm

Algorithm 2 merges a bracket of LDR images to create an HDR image. As mentioned in the main paper, we merge in the luminance space to avoid ghosting artifacts. Our luminance conversion assumes sRGB [42] and gamma value of 2.4, following [59].

Algorithm 1: Iterative Inpainting Algorithm

function INPAINTBALL($\mathbf{I}, \mathbf{M}, \mathbf{C}, \gamma, T = 1000$)
  $\mathbf{z} \leftarrow$ ENCODE$(\mathbf{I})$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_{\gamma T} \leftarrow \sqrt{\alpha_{\gamma T}}\mathbf{z} + \sqrt{1-\alpha_{\gamma T}}\epsilon$
  return INPAINT$(\mathbf{z}, \mathbf{z}_{\gamma T}, \mathbf{M}, \mathbf{C}$, denoising-start $= \gamma T)$
end function

// **input:** Input image $\mathbf{I}$, binary inpainting mask $\mathbf{M}$,
// Conditioning signal (e.g. text, depth map) $\mathbf{C}$
// denoising strength γ, number of balls for median N,
// number of median iterations k.
// **output:** Input image with an inpainted chrome ball.
function ITERATIVEINPAINT($\mathbf{I}, \mathbf{M}, \mathbf{C}, \gamma, k, N$)
  for $i \in \{1, \ldots, k\}$ **do**
    for $j \in \{1, \ldots, N\}$ **do**
      $\gamma' \leftarrow \gamma$ **if** $i > 1$ **else** 1.0
      $\mathbf{B}_j \leftarrow$ INPAINTBALL$(\mathbf{I}, \mathbf{M}, \mathbf{C}, \gamma')$
    end for
    $\mathbf{B} \leftarrow$ PIXELWISEMEDIAN$(\mathbf{B}_1, \ldots, \mathbf{B}_N)$
    $\mathbf{I} \leftarrow (1-\mathbf{M}) \odot \mathbf{I} + \mathbf{M} \odot \mathbf{B}$
  end for
  return INPAINTBALL$(\mathbf{I}, \mathbf{M}, \gamma)$
end function

Algorithm 2: HDR Merging Algorithm

function LUMINANCE($\mathbf{I}, ev, \gamma = 2.4$)
  return $\mathbf{I}^\gamma \cdot [0.21267, 0.71516, 0.07217]^\top (2^{-ev})$
end function

// **input:** LDR images and a list of exposure values in
// descending order, where $ev_0 = 0$. E.g., [0, -2.5, -5]
// **output:** A linearized HDR image.
function MERGELDRS($\mathbf{I}_0, ..., \mathbf{I}_{n-1}, ev_0, ..., ev_{n-1}$)
  $\mathbf{L} \leftarrow$ LUMINANCE$(\mathbf{I}_{n-1}, ev_{n-1})$
  for $i \in \{n-2, n-1, ..., 0\}$ **do**
    $\mathbf{L}_i \leftarrow$ LUMINANCE$(\mathbf{I}_i, ev_i)$
    $\mathbf{M} \leftarrow$ CLIP$(\frac{(2^{ev_i}\mathbf{L}_i)-0.9}{0.1}, 0, 1) \odot \mathbb{1}(\mathbf{L} > \mathbf{L}_i)$
    $\mathbf{L} \leftarrow (1-\mathbf{M}) \odot \mathbf{L}_i + \mathbf{M} \odot \mathbf{L}$
  end for
  return $\mathbf{I}_0^\gamma \odot \left(\frac{\mathbf{L}}{\mathbf{L}_0}\right)$
end function

B. Ablation on Iterative Inpainting Algorithm

B.1. Inpainting iterations

Figure 10 presents additional results demonstrating how our iterative inpainting algorithm improves the consistency and quality of the generated chrome balls after one iteration. Figure 11 presents results with more iterations. Note that the experiments in our main paper were limited to two iterations due to our resource constraints.

B.2. Trade-off between running time and quality

We experimented with various numbers of balls N and iterations k in the iterative inpainting algorithm on the validation set comprising 100 scenes from the Laval Indoor dataset

#Iterations (k)	#Balls (N)	Time (mins)	si-RMSE ↓	Angular Error ↓	Normalized RMSE ↓
1	5	2.5	0.631	5.367	0.416
	15	7.5	0.615	5.229	0.410
	30	15	0.609	5.210	0.405
2	5	5	0.634	5.393	0.417
	15	15	0.615	5.263	0.409
	30	30	0.607	5.248	0.403

Table 5. Ablation study on the number of iterations k and balls N. We report running time and quality metrics on 200 mirror balls in the validation set. See Figure 9 for qualitative results.

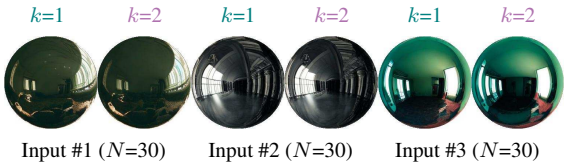


Figure 9. Qualitative results for the two configurations in Table 5, marked with teal and purple colors. We can halve the running time with minimal quality degradation by decreasing the number of iterations k to one.

[20] and 100 scenes from the Poly Haven dataset [3]. As shown in Table 5, increasing either of them generally leads to more accurate results. Notably, while two iterations of iterative inpainting ($k = 2$) deliver the best score, reducing the number of iterations to one ($k = 1$) halves the running time with minimal quality degradation as shown in Figure 9. Note that we opted for $N = 30$ and $k = 2$, which takes about 30 minutes per image on an RTX 3090 Ti GPU, because of resource constraints.

B.3. Inpainting ball size

We investigated the effects of ball diameter (i.e., white circle) on the depth maps used by ControlNet. Specifically, we analyze and compare the efficacy of various ball sizes, ranging from 128 to 512 pixels in diameter, as illustrated in Figure 12. The result shows that increasing the ball size from 256 to 384 or 512 still results in realistic balls, but they do not reflect the environment as convincingly. This is likely because the original input content seen by the model is reduced. On the other hand, smaller balls can capture the lighting well but are less detailed and not as useful.

C. Ablation on LoRA Training

C.1. Additional details on training set generation

To generate training panoramas from text prompts as mentioned in Section 3.3, we use Text2Light [10], using prompts from its official GitHub repository[‡] and additional prompts generated by Chat-GPT 3.5[§] from short instructions and examples. To eliminate near-duplicate samples, we use the perceptual hashing algorithm implemented in the `imagededup` package[¶]. This process yielded a dataset of 1,412 unique HDR panoramas at resolution of 2048×4196 pixels. We used orthographic projection and a 60° field of view.

[‡] https://github.com/FrozenBurning/Text2Light

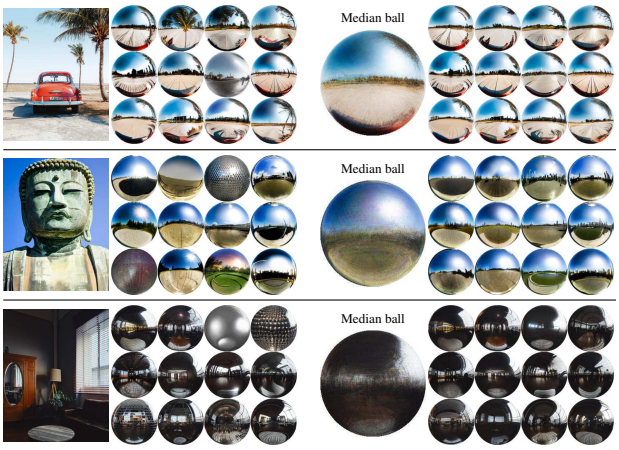


Figure 10. Chrome balls before (left) and after (right) one iteration of our iterative inpainting algorithm. Notice how poor chrome balls are fixed and the light estimation becomes more consistent.

C.2. Range of timesteps for LoRA training

For LoRA training, we sampled from timesteps 900-999 as we observed that the overall lighting information is determined earlier in the sampling process (see Figure 13). This choice helped speed up training. In Table 6, we compare this choice to training from 0-999 and 500-999 given the same number of training iterations and report scores on the same validation set of 200 scenes as in Appendix B.2. Our choice of 900-999 yielded the best performance across all three metrics.

C.3. LoRA scale

We conducted an experiment to assess the effect of using different LoRA scales. Here, the LoRA scale refers to the α value in the weight update equation: $\mathbf{W}' = \mathbf{W} + \alpha\Delta\mathbf{W}$, where $\mathbf{W}'$ is the new weight for inference, $\mathbf{W}$ is the original weight of SDXL, $\Delta\mathbf{W}$ is the weight update from LoRA. (See Section 3.1 for a brief background on LoRA.) In Table 7, we report scores computed on EV0 LDR chrome balls evaluated on scenes in Poly Haven, which were never part of Text2Light's training set. We selected the LoRA scale of 0.75, which has the best si-RMSE and angular error scores, for our implementation.

[§] https://chat.openai.com/
[¶] https://github.com/idealo/imagededup

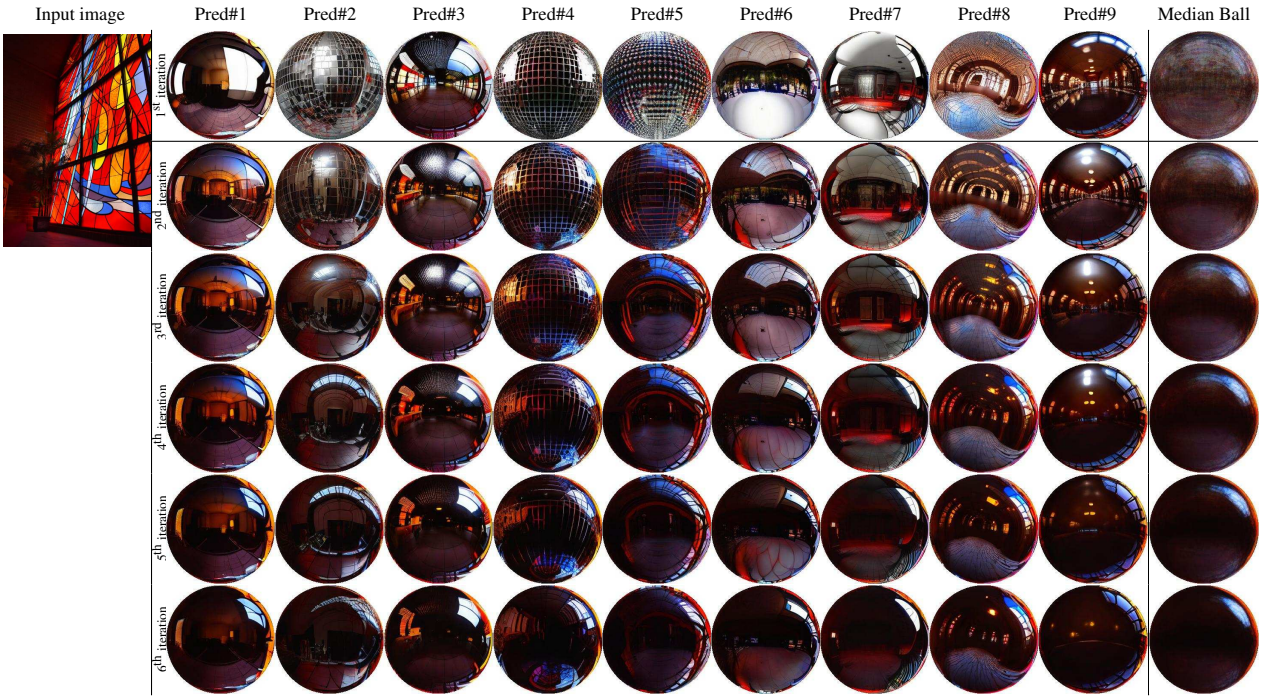


Figure 11. Repeatedly applying our iterative inpainting algorithm gradually produces chrome balls with better light estimation and fix degenerate balls such as Pred#2, Pred#4, and Pred#5.

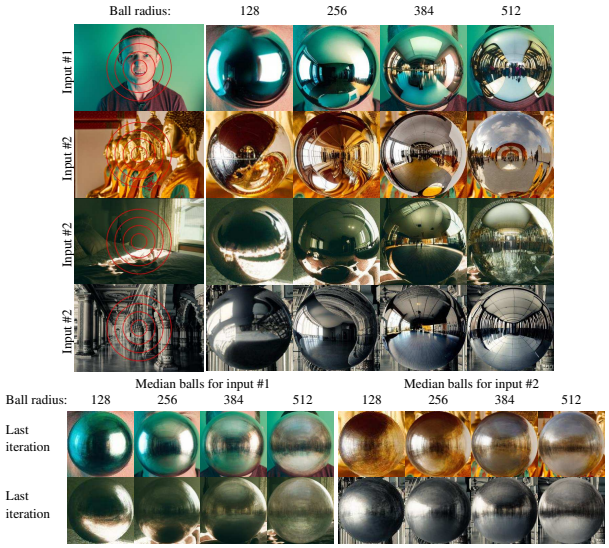


Figure 12. Results when varying the ball size (without LoRA).

C.4. Training a single continuous LoRA v.s. multiple LoRAs for exposure bracketing

As described in Section 3.3, we train a single *continuous* LoRA for multiple EVs by conditioning it on an interpolated text prompt embedding instead of training multiple LoRAs for individual EVs. This strategy helps preserve the overall scene structure across exposures due to weight sharing.

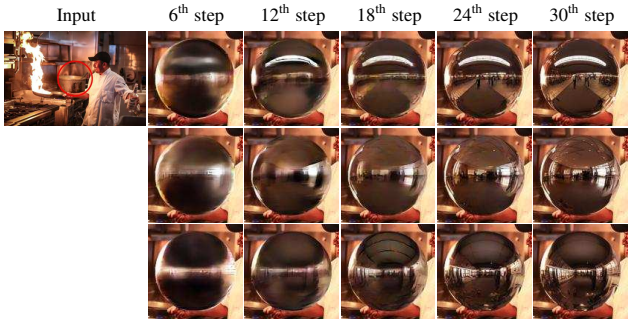


Figure 13. The overall lighting is determined at early sampling steps. Here, we visualize intermediate predictions at various steps during 30-step sampling with UniPC [70]. These intermediate predictions, or the predicted $\mathbf{z}_0$, can be computed from $\mathbf{z}_t$ at any timestep t using Equation 1. Each row corresponds to a different random seed.

To show this, we conducted an experiment comparing results from our LoRA and three separately trained LoRAs at EVs 0, -2.5, and -5.0. Following the commonly adopted training pipeline [51] implemented in the Diffusers library [55], our three LoRAs are trained with the prompt containing the '*sks*' token: "a perfect sks mirrored reflective chrome ball sphere." We use the same hyperparameters, random seeds, and HDR panoramas during training. We present results without our iterative algorithm to isolate the effect of LoRA in Figure 14. Note that we use a lora scale of 1.0 to apply

Figure 14. Our proposed continuous LoRA training (Cont. LoRA) yields chrome balls with higher detail consistency across different EVs than results obtained from using three separate LoRAs (3 LoRAs).

Sphere	Denoising step	si-RMSE ↓	Angular Error ↓	Normalized RMSE ↓
Diffuse	$\mathbf{x}_0 : \mathbf{x}_{999}$	0.194	3.322	0.292
	$\mathbf{x}_{500} : \mathbf{x}_{999}$	0.188	3.260	0.284
	$\mathbf{x}_{900} : \mathbf{x}_{999}$	0.156	2.956	0.246
Matte	$\mathbf{x}_0 : \mathbf{x}_{999}$	0.449	4.121	0.436
	$\mathbf{x}_{500} : \mathbf{x}_{999}$	0.452	4.008	0.438
	$\mathbf{x}_{900} : \mathbf{x}_{999}$	0.385	3.575	0.371
Mirror	$\mathbf{x}_0 : \mathbf{x}_{999}$	0.727	6.292	0.483
	$\mathbf{x}_{500} : \mathbf{x}_{999}$	0.730	6.277	0.479
	$\mathbf{x}_{900} : \mathbf{x}_{999}$	0.656	5.464	0.431

Table 6. Ablation study on sampled timesteps for LoRA training.

LoRA scale	RMSE ↓	si-RMSE ↓	Angular Error ↓
0.00	0.232	0.327	6.189
0.25	0.220	0.307	6.287
0.50	0.211	0.303	6.180
0.75	0.204	0.300	6.109
1.00	0.199	0.303	6.267

Table 7. Ablation study on LoRA scales

the same weight residual matrices obtained from the training. Our LoRA produces chrome balls with better structure consistency, particularly at EV-5.0.

D. More Comparison with SOTA Inpainting Techniques

In Figure 2 in Section 2, we provide a qualitative comparison between our approach and existing SOTA diffusion-based inpainting methods: Blended Diffusion [6, 7], Paint-by-Example [63], IP-Adapter [64], DALL·E2 [1], Adobe Firefly [2], and SDXL [45]. In this section, we describe the experimental settings for these methods. Additionally, we investigate the behavior of each using different random seeds.

D.1. Experimental setups

Blended Diffusion, IP-Adapter, and SDXL shared the same text prompt: "a perfect mirrored reflective chrome ball sphere.". We used negative prompt "matte, diffuse, flat, dull" when executing methods that can accept one: IP-Adapter and SDXL. We provided Paint-by-Example and IP-Adapter with reference chrome balls from five randomly selected HDR environment maps in Poly Haven dataset [3] as shown in Figure 15. We used the official OpenAI API [‖] for DALL·E2, and we used the Generative Fill feature in Photoshop for Adobe Firefly. We followed the default configurations in the methods' official implementations as described in Table 8.

D.2. Behavior under different random seeds

We show inpainting results of our method and other baselines using different random seeds in Figure 21 and Figure

[‖] https://platform.openai.com/docs/guides/images/edits-dall-e-2-only

Method	Sampler	#step	cfg
Blended Diffusion	DDIM [26]	50	7.5
Paint-by-Example	PLMS [35]	50	5.0
IP-Adapter	UniPC [70]	30	5.0
SDXL	UniPC [70]	30	5.0

Table 8. Sampler, number of sampling steps, and classifier-guidance scale (cfg) used in different SOTA methods.

Figure 15. Chrome balls used as inputs for Paint-by-Example [63] and IP-Adapter [64]. We generate them from five randomly selected HDR environment maps from Poly Haven dataset [3].

22. What we observed in general was that Blend Diffusion [6, 7] produced distorted balls. Paint-by-Example [63] failed to reproduce mirrored chrome balls altogether. IP-Adapter [64] replicated textures and details of the example chrome balls, making it unsuitable for light estimation. DALL·E2 [1] often simply reconstructed most of the masked-out content. Adobe Firefly [2] had a similar problem with DALL·E2 [1], albeit more severe (see Figure 22). Moreover, none of these techniques precisely followed the inpainting mask. Our proposed method can address all these issues and consistently inpaint high-quality chrome balls.

E. Additional Qualitative Results

E.1. Benchmark datasets

This section provides qualitative results for the experiments in Section 4 in the main paper.

Evaluation on three spheres. We show spheres with three material types—mirror, matte, and diffuse—rendered using the inferred environment maps from the following methods:

1. The ground truth
2. StyleLight [59]
3. Stable Diffusion XL [45] with depth-conditioned ControlNet [69] (SDXL†)
4. SDXL† with our LoRA (SDXL†+LR)
5. SDXL† with our iterative inpainting (SDXL†+I)
6. SDXL† with our LoRA and iterative inpainting (SDXL†+LR+I)

Qualitative results for the Laval indoor dataset are in Figure 26-25 and for Poly Haven in Figure 27-29. As discussed in the main paper, we start with SDXL† to which we add our LoRA (LR) and iterative inpainting (I) to obtain our proposed algorithm, SDXL†+LR+I. The methods designated SDXL†+LR and SDXL†+I are ablated versions of our algorithm.

Evaluation on an array of spheres. We show renderings of an array of spheres on a plane using our estimated lighting, following the evaluation protocol from Weber et al. [62]. We display 24 random test images from a total of 2,240 test images of the Laval Indoor dataset in Figure 31. Because the scale of the HDR images our method generates and that of the test set are different, for visualization purposes, we scale each output image so that its 0.1^{st} and 99.9^{th} percentiles of pixel intensity match those of the ground truth. Note that this scaling does not affect the quantitative scores reported in the main paper since the metrics are already scale-invariant. The last row shows challenging test scenes featuring only plain, solid backgrounds without any shaded objects. Estimating lighting from such input images is highly ill-posed and multimodal. As a result, visual assessment or evaluations using pixel-based metrics, as used in this protocol, may not be meaningful for such cases.

E.2. In-the-wild images

We present additional qualitative results for in-the-wild scenes in Figure 32. Our method produces high-quality chrome balls that harmonize well with diverse scenes and lighting environments, such as a dim hallway under red neon lighting, an underwater tunnel with blue-tinted sunlight, a close-up shot of food, an outdoor view by a coastline, and a bird's-eye view from a tall building. Our method also works on non-realistic images, such as paintings or cartoons, where visual cues like shading and shadows are present (see Figure 33).

F. Stochastic vs Deterministic Sampling

In Section 3.2, we discuss the relationship between initial noise maps and semantic patterns of chrome balls. This mapping is deterministic only when using samplers derived from probability flow ODE, such as DDIM [53] and UniPC [70]. The "disco" noise map would less consistently produce "disco" balls if we adopted stochastic samplers such as DDPM [26], which introduce noise during the sampling process. This section investigates whether degeneration of chrome balls is caused by deterministic sampling and whether stochastic sampling can help mitigate this issue.

We conducted an experiment comparing results from DDIM and DDPM using different numbers of sampling steps. Our results suggest that neither of these schemes consistently reduces occurrence of bad chrome balls. Specifically, using stochastic samplers may occasionally produce better chrome balls than deterministic ones at sufficiently high numbers of sampling steps (see Figure 16). Unfortunately, they still yield "disco" balls when starting sampling with the "disco" noise map, as illustrated in Figure 17.

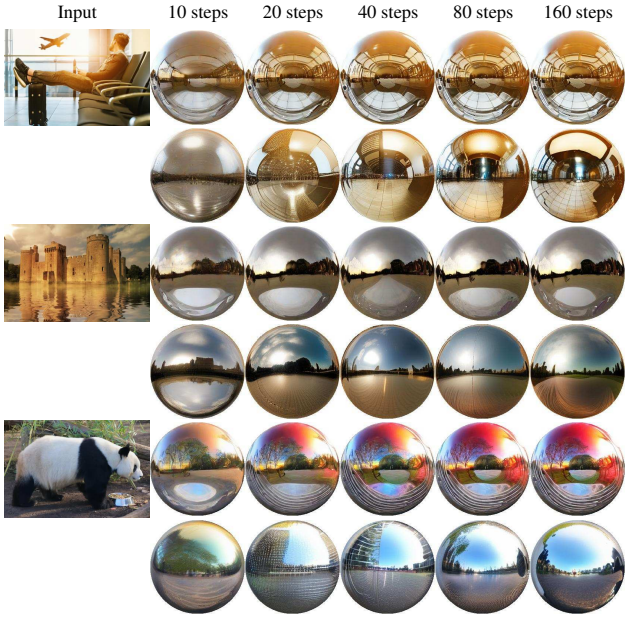


Figure 16. Comparison between chrome balls generated using DDIM [53] (1st row) and DDPM [26] (2nd row) with different sampling steps. DDPM can sometimes mitigate the occurrence of bad patterns originating from bad initial noise maps when using high sampling steps. Prompt: "a chrome ball".

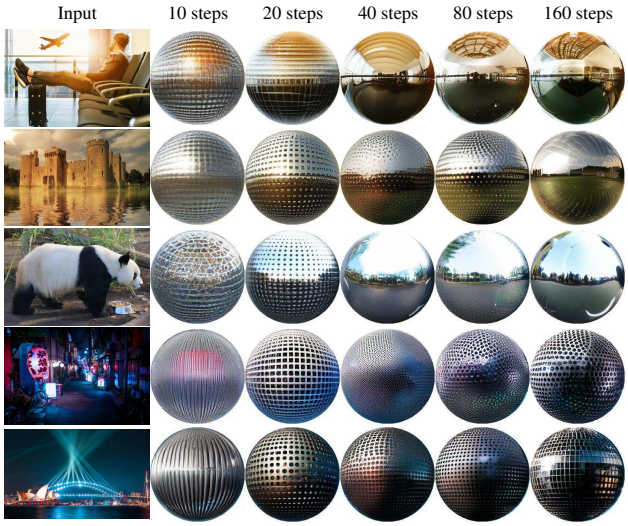


Figure 17. Chrome balls generated from the "disco" noise map using DDPM [26] with different sampling steps. Switching to DDPM instead of deterministic samplers still produces "disco" balls, even at high sampling steps. Prompt: "a chrome ball".

G. Spatially-Varying Light Estimation

In this work, we inpaint a chrome ball in the input's center to represent global lighting in the scene and do not model any spatially varying effects by assuming orthographic projection. Nonetheless, our preliminary study suggests that the output from our inpainting pipeline does change according to where the chrome ball is inpatined, as illustrated in Figure 18. This behavior can be leveraged for spatially-varying light estimation. To correctly infer spatially-varying, omnidirectional lighting, one needs to also infer the scene geometry, the depth of the inpainted chrome ball and camera parameters such as the focal length from the input image. These problems are interesting areas for future work.

H. Virtual Object Insertion

Virtual object insertion is a downstream application that requires light estimation. In Figure 19, we present qualitative results for two objects from Objaverse-XL [14], rendered with HDR environment maps obtained through our method.

I. Additional Failure Cases

We present additional failure cases in Figure 20. Our method occasionally fails to produce chrome balls that accurately reflect surrounding environments in overhead or bird's-eye view images. For instance, the curvature of the horizon line in the scene with balloons is incorrect. While our method performs reasonably well for some non-realistic images like paintings, it struggles with images in drastically different styles, such as some cartoons and Japanese-style animations, which significantly differ from the training data of SDXL [45]. Switching to a fine-tuned model, such as AnimagineXL [**], to leverage a more specialized generative prior can improve its performance on specific image styles.

J. StyleLight's Score Discrepancies

We used StyleLight's official implementation to produce their scores in Table 1. However, there are discrepancies with their reported scores due to unknown implementations of their metrics. We discussed this with the authors on GitHub [‡‡] before the submission deadline, and they clarified that additional masking of black regions and rotation of panoramas were performed before evaluation. Despite implementing these additional steps, we still could not match the scores. The authors further suggested that we apply a consistent post-processing technique to all baselines for a fair comparison, which resulted in the scores we reported. To ensure transparency, we have made our evaluation code available at https://github.com/DiffusionLight/DiffusionLight-evaluation.

[**] https://huggingface.co/Linaqruf/animagine-xl
[††] https://www.kaggle.com/datasets/mylesoneill/tagged-anime-illustrations/data
[‡‡] https://github.com/Wanggcong/StyleLight/issues/9

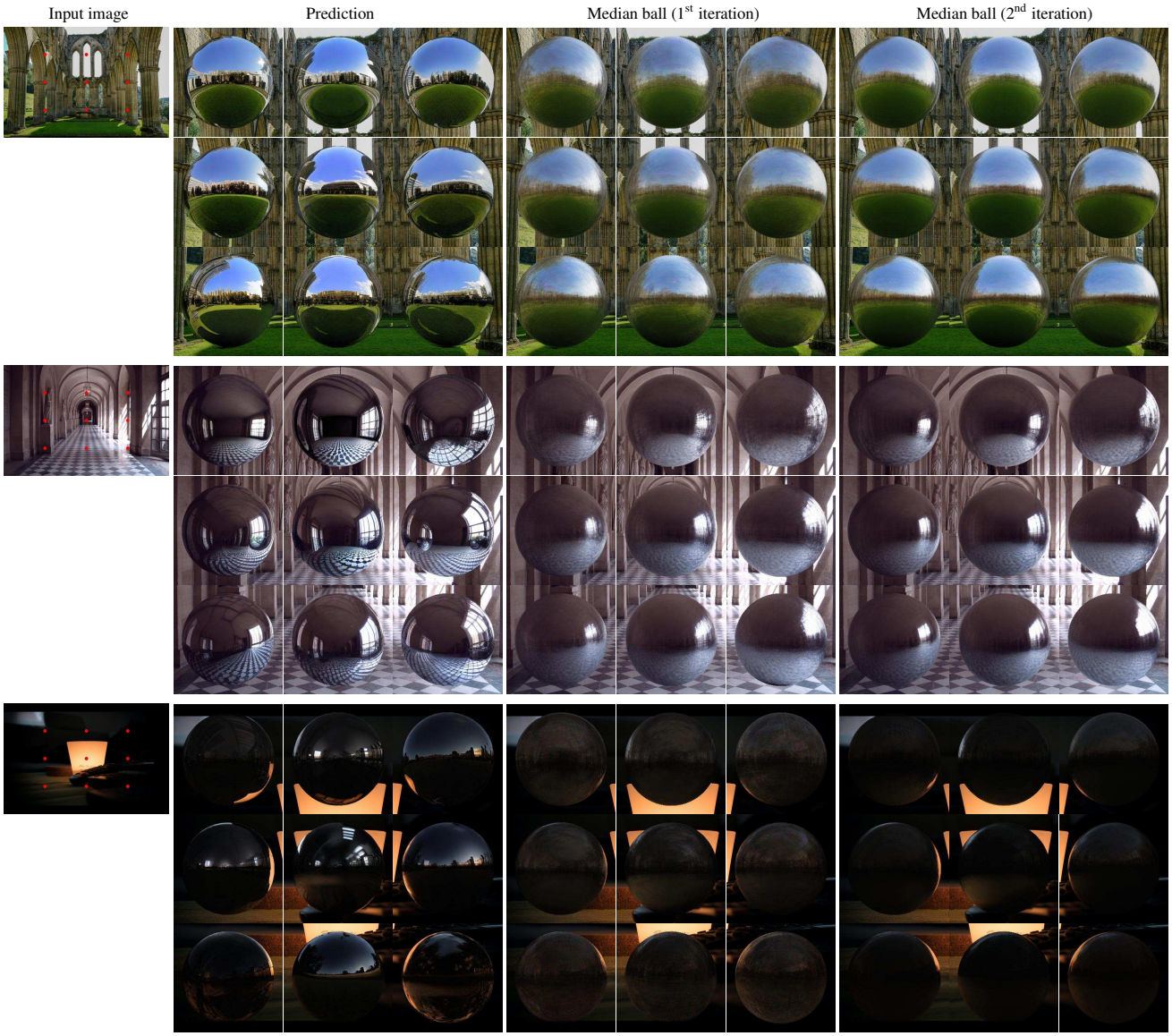
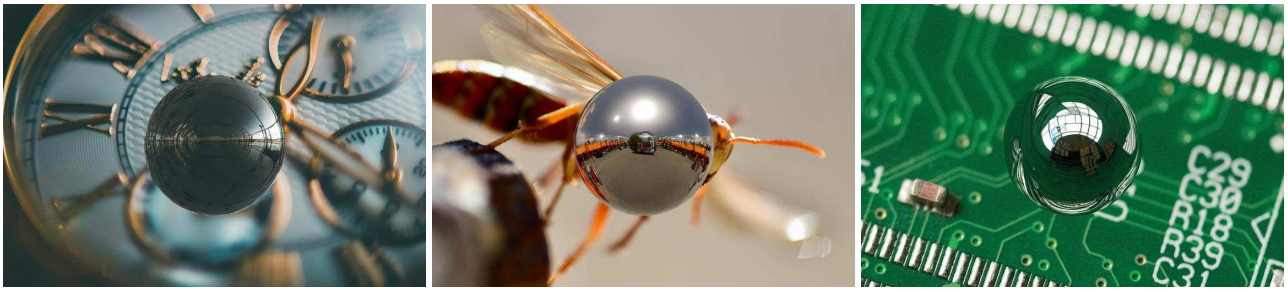


Figure 18. We show the spatially varying effects of painting a chrome ball at nine different locations, specified by red dots in the input images. For each input image, we present predictions from a random seed and median balls at the 1st and 2nd iterations. The effects can be seen in the changes of the curvature the horizon line, the size of the window, and the position of the light reflected from the lamp. These effects are more apparent in median balls as the number of iterations increases.

Figure 19. We synthetically render each 3D object into input images using our estimated lighting.

(a) Overhead and bird's-eye view images

(b) Images with significant style difference from natural photos



(c) Marcro and close-up images

(d) High-key and low-key images with solid backgrounds

Figure 20. **Failure modes**: (a) Our method may produce unrealistic chrome balls in overhead and bird's-eye view images, leading to incorrect curvature in the horizon line. (b) The chrome balls may not harmonize well with inputs whose styles differ significantly from natural photos. (c) Macro and close-up images can lack sufficient shading cues, leading to less convincing estimates. (d) Images with empty or solid backgrounds often cause our method to hallucinate some details onto the balls, and the balls may appear too dark, especially on a white background. These images are from Unsplash (www.unsplash.com), Kaggle [††], or other websites under the CC 4.0 license.

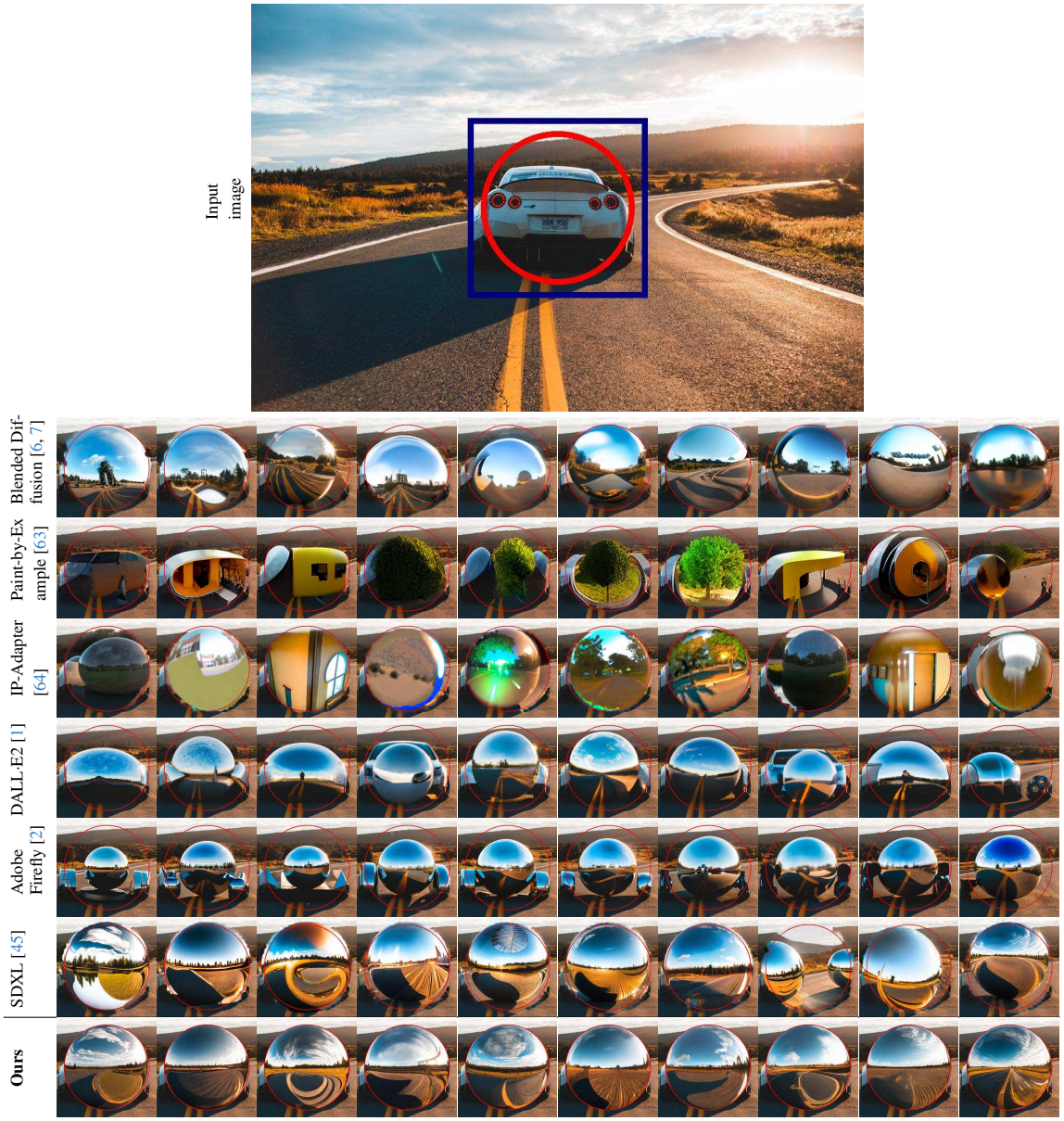


Figure 21. Chrome ball inpainting results from various methods. The red circle indicates the inpainted region, and we show a zoomed-in view of the blue crop. Each row contains results from ten different random seeds.

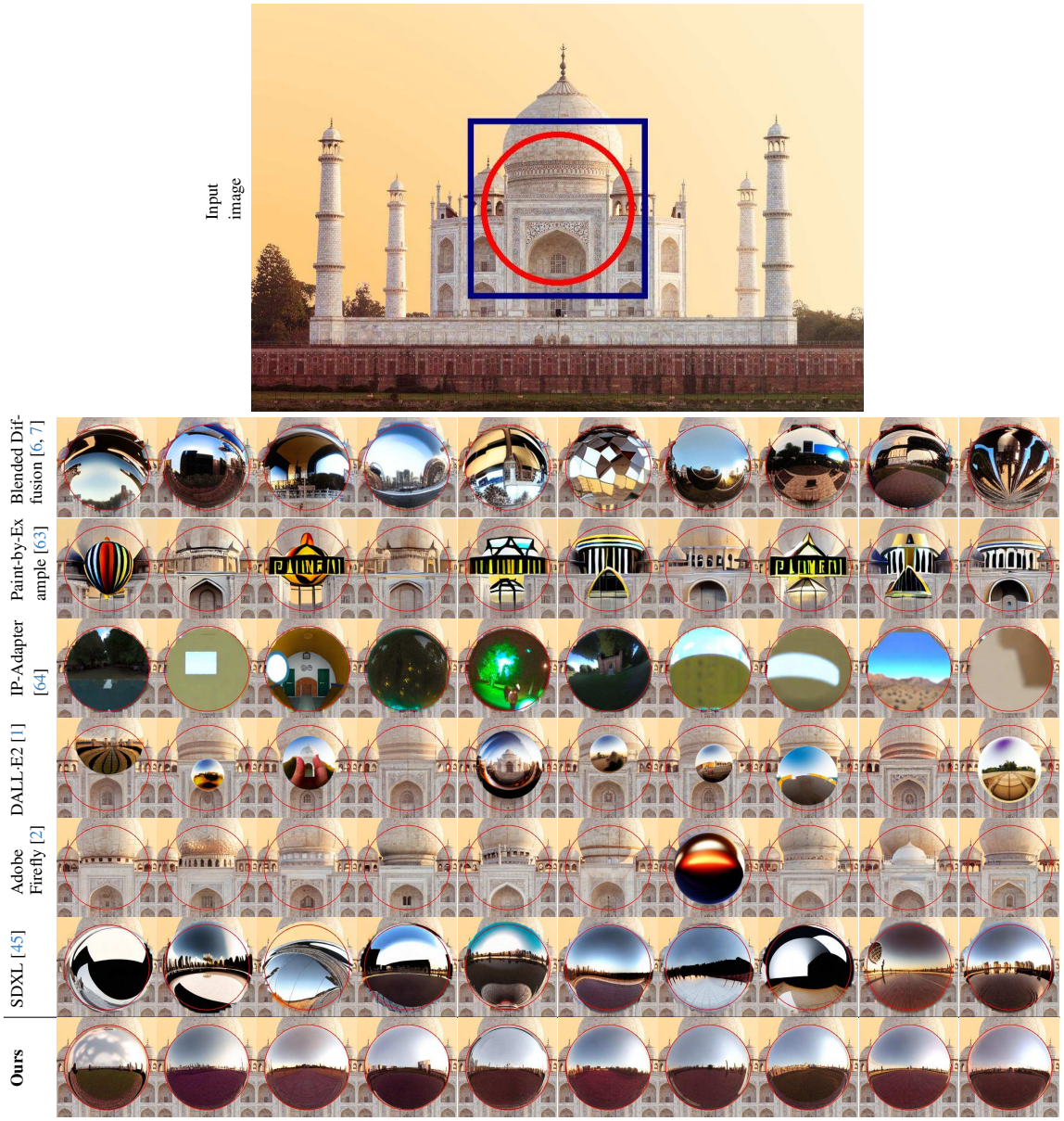


Figure 22. Chrome ball inpainting results from various methods. The red circle indicates the inpainted region, and we show a zoomed-in view of the blue crop. Each row contains results from ten different random seeds.

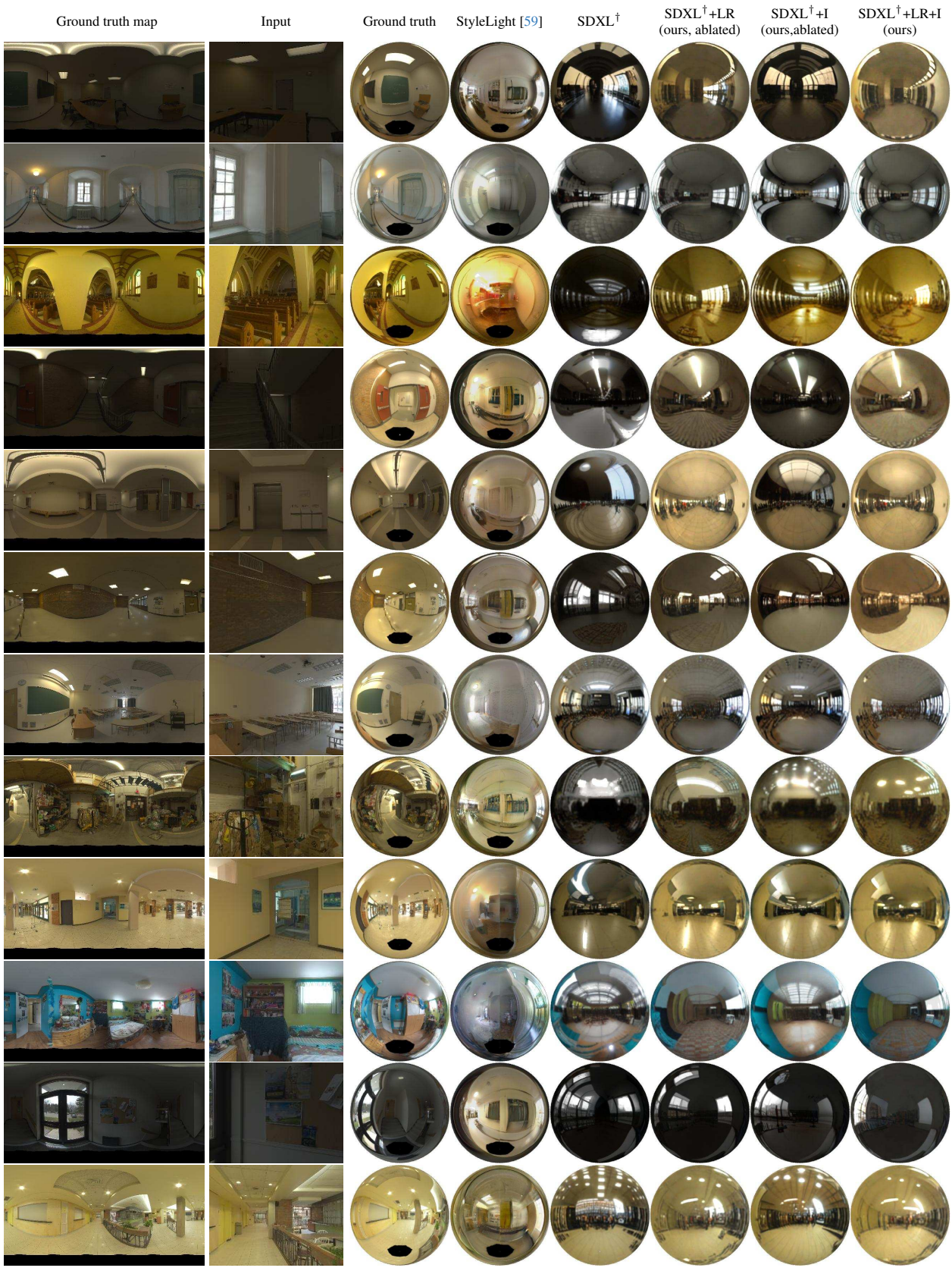


Figure 23. Qualitative results for the Laval indoor dataset using mirror balls.

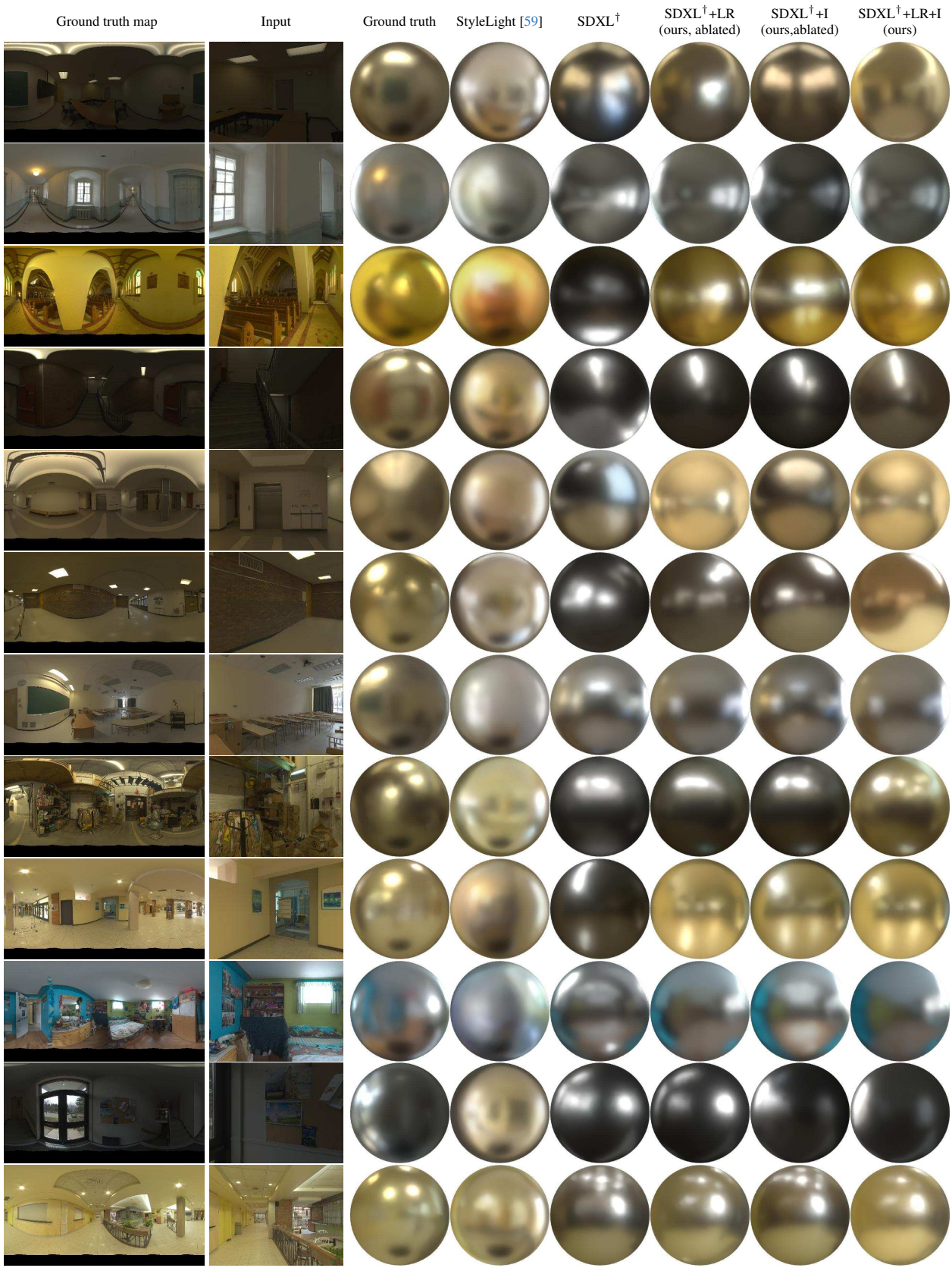


Figure 24. Qualitative results for the Laval indoor dataset using matte balls.

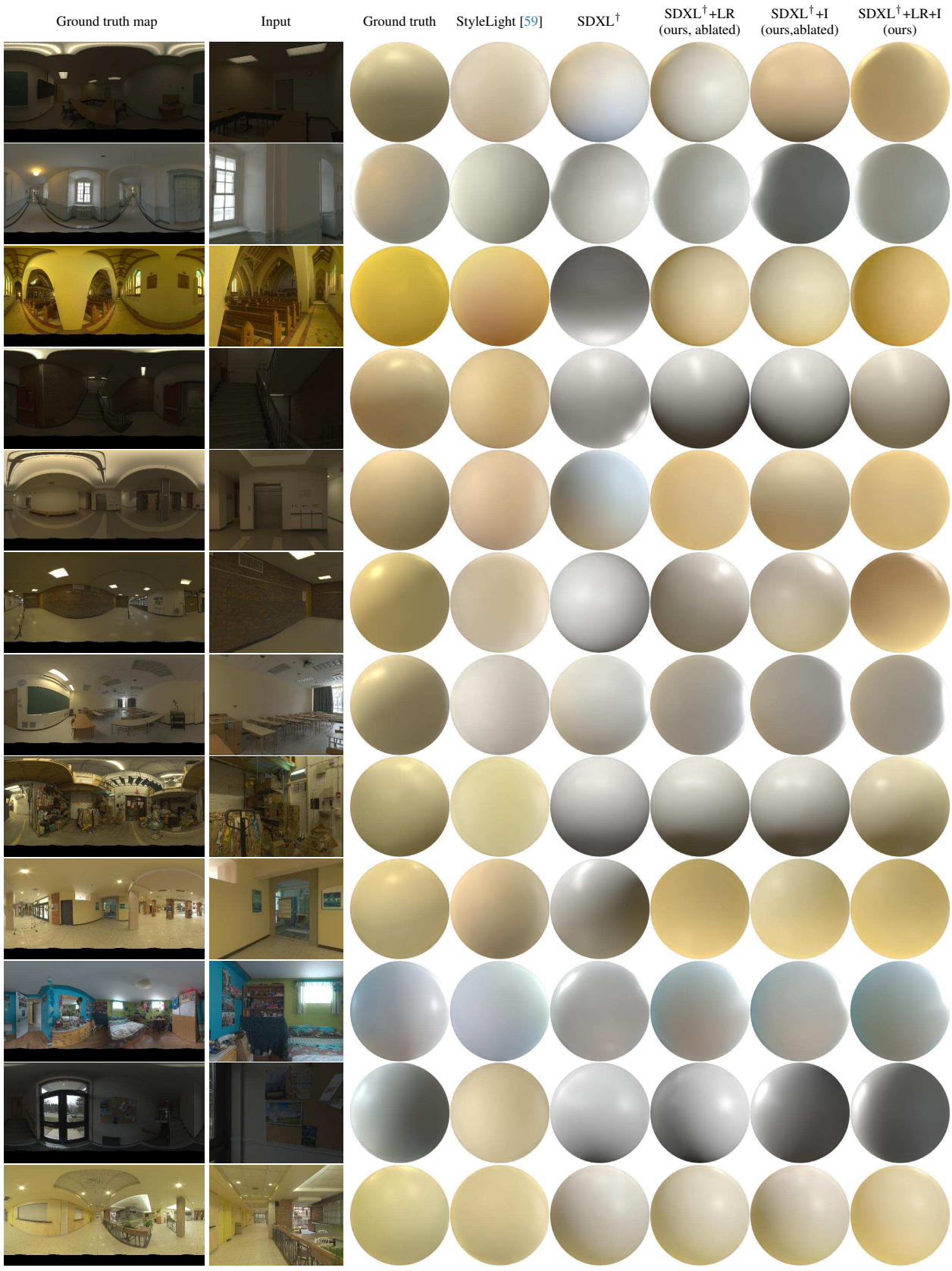


Figure 25. Qualitative results for the Laval indoor dataset using diffuse balls.

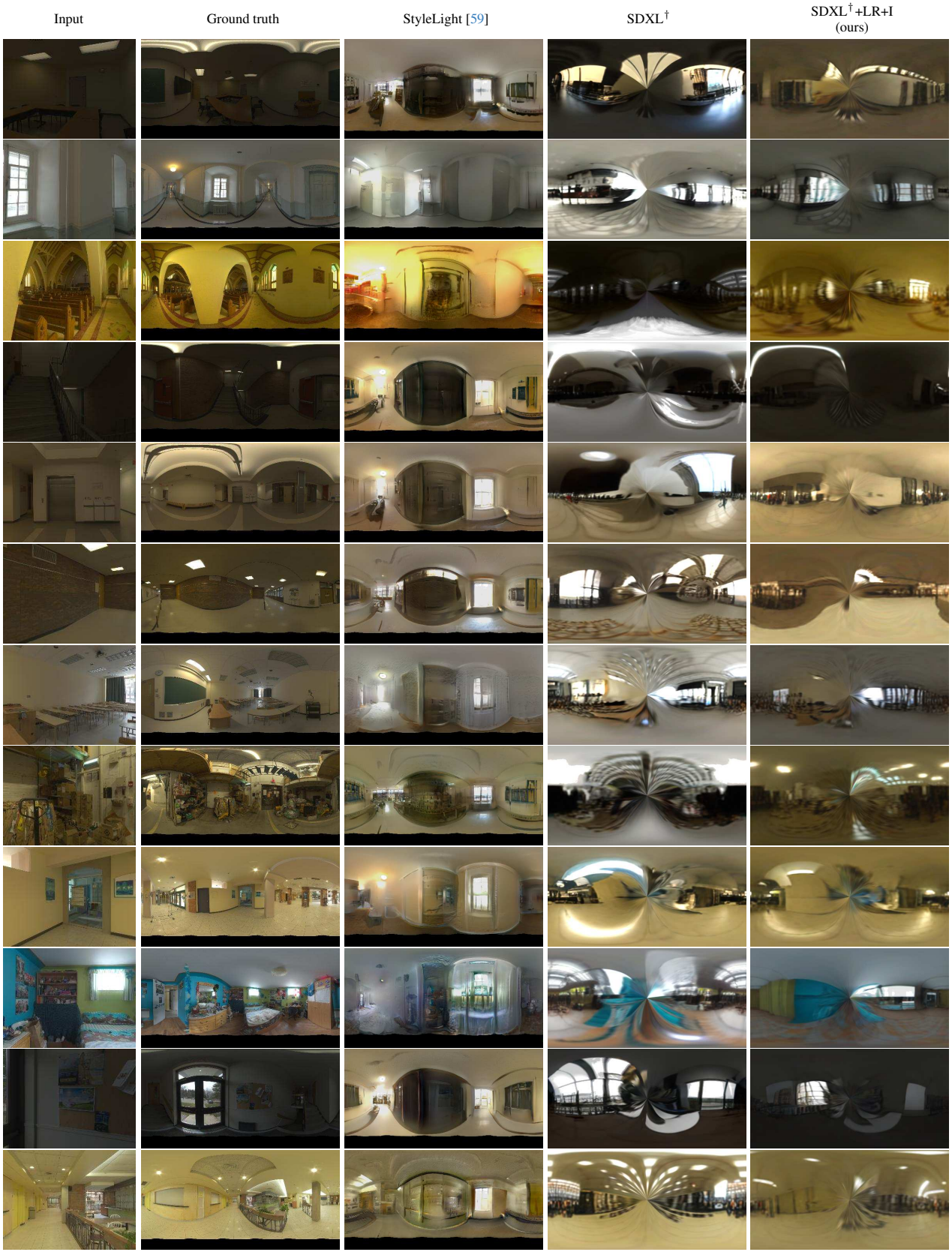


Figure 26. Unwarped equirectangular maps for the Laval indoor dataset.

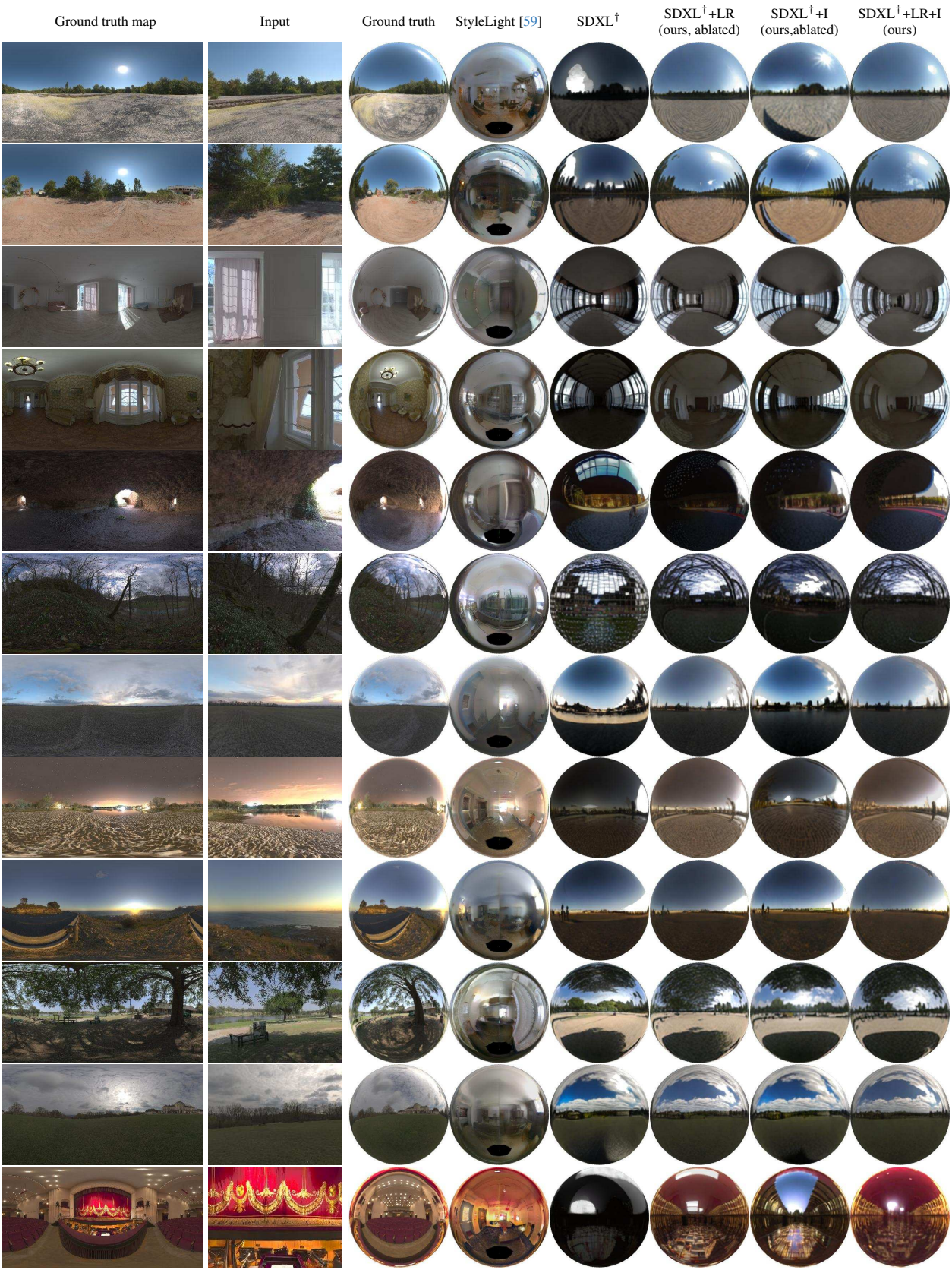


Figure 27. Qualitative results for the Poly Haven dataset using mirror balls.

Figure 28. Qualitative results for the Poly Haven dataset using matte balls.

Figure 29. Qualitative results for the Poly Haven dataset using diffuse balls.

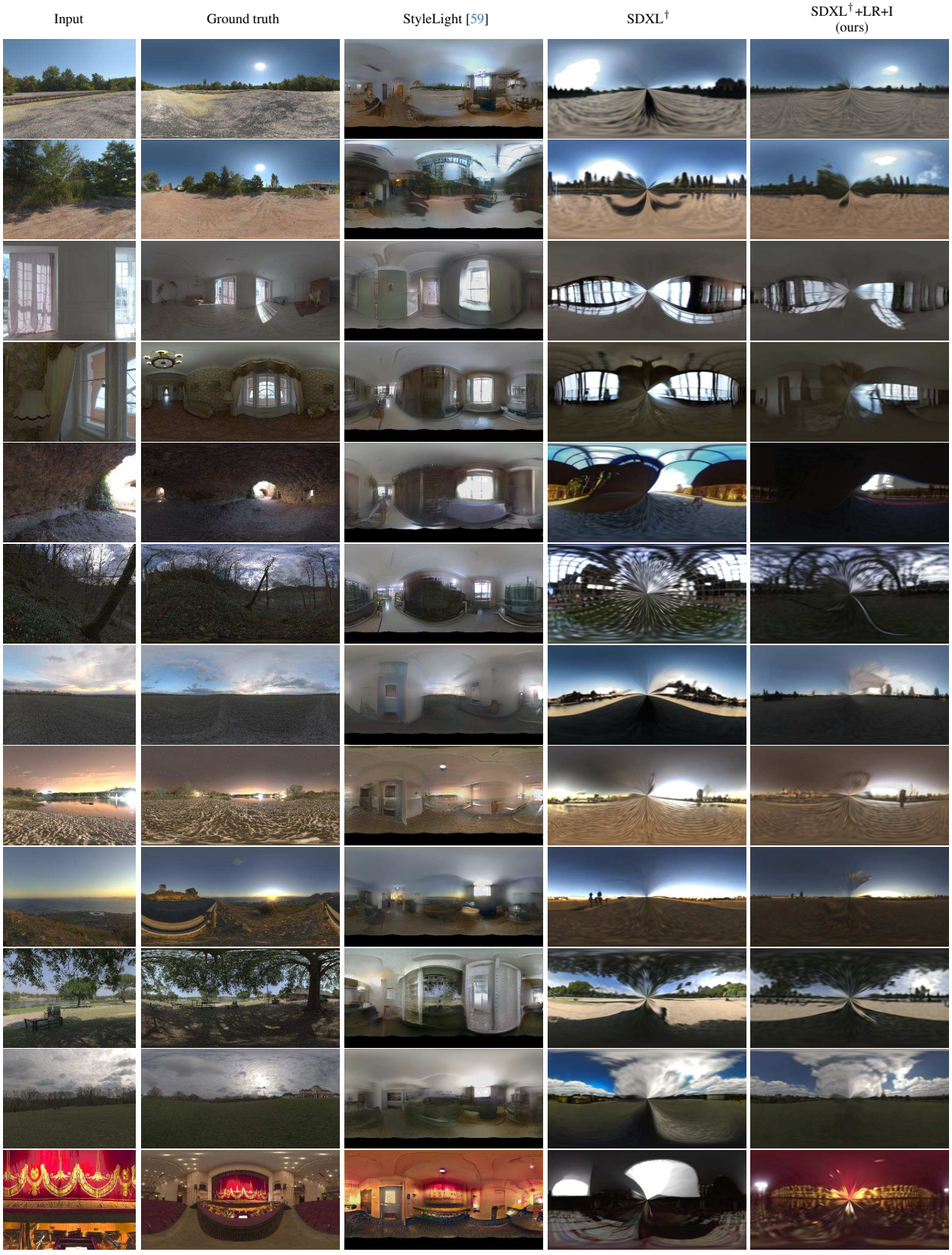


Figure 30. Unwrapped equirectangular maps for the Poly Haven dataset.

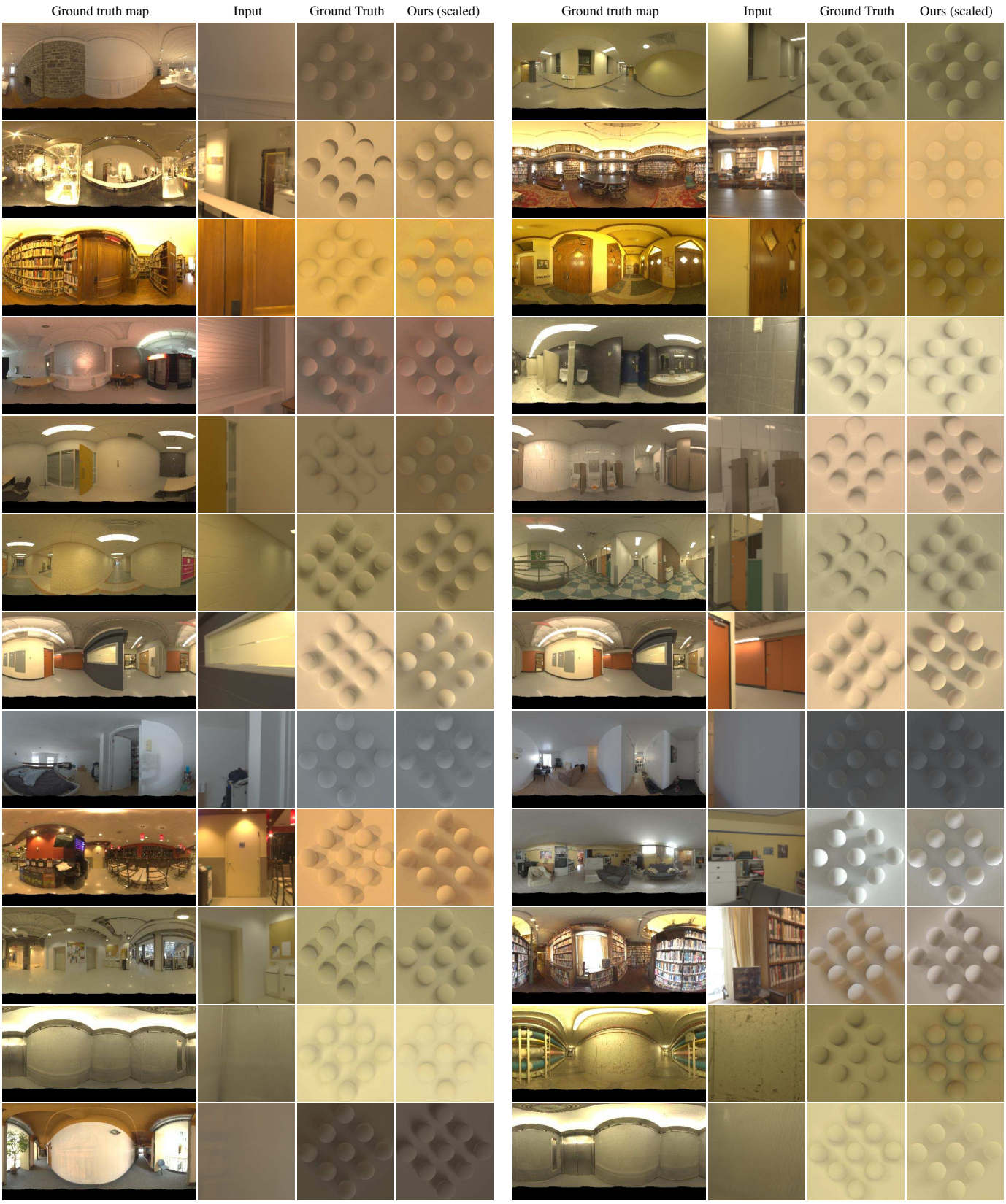


Figure 31. Qualitative results for the Laval indoor dataset using an array of spheres.

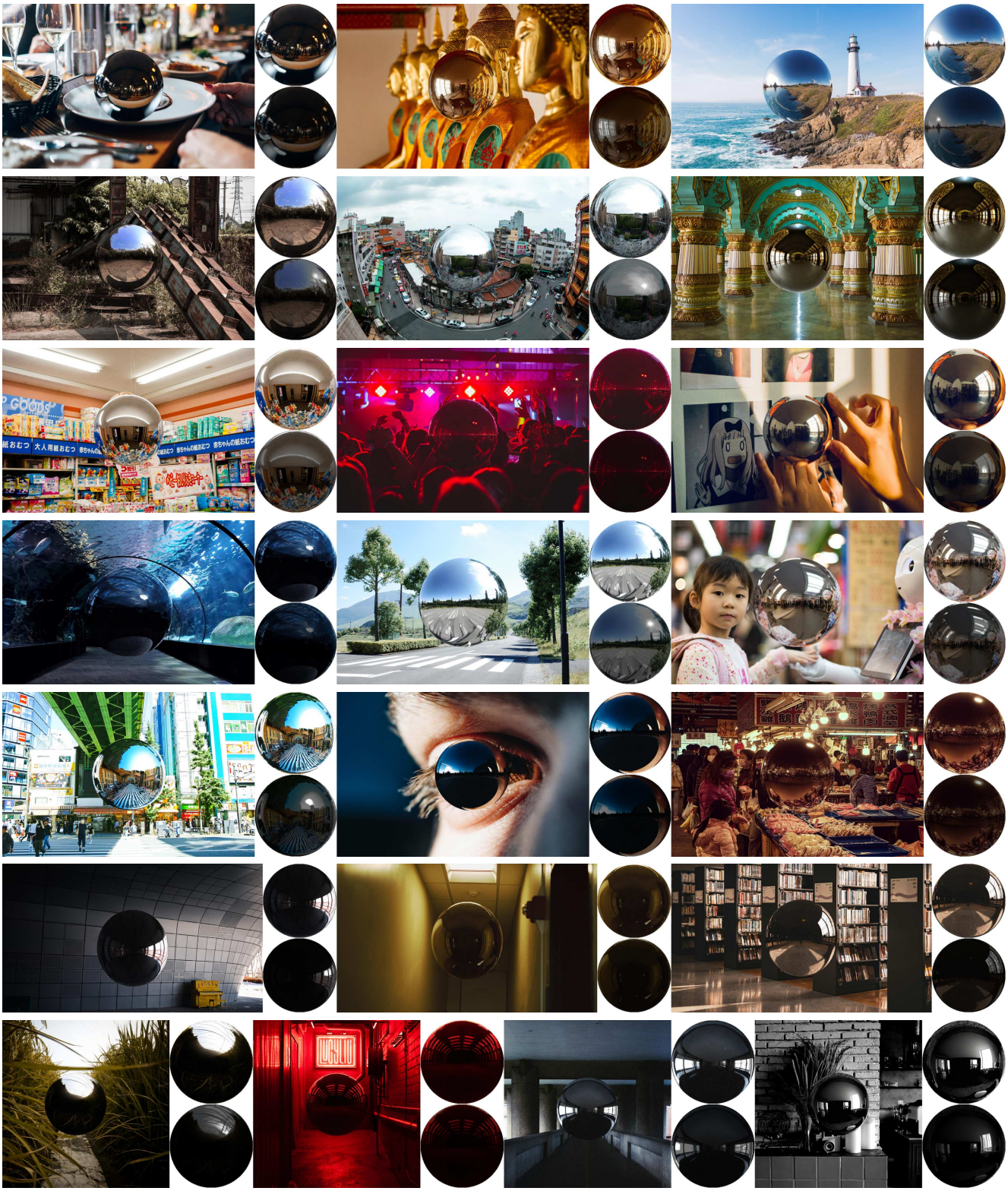


Figure 32. Additional qualitative results for in-the-wild scenes. For each input, we show a chrome ball generated from our pipeline and its underexposed version.

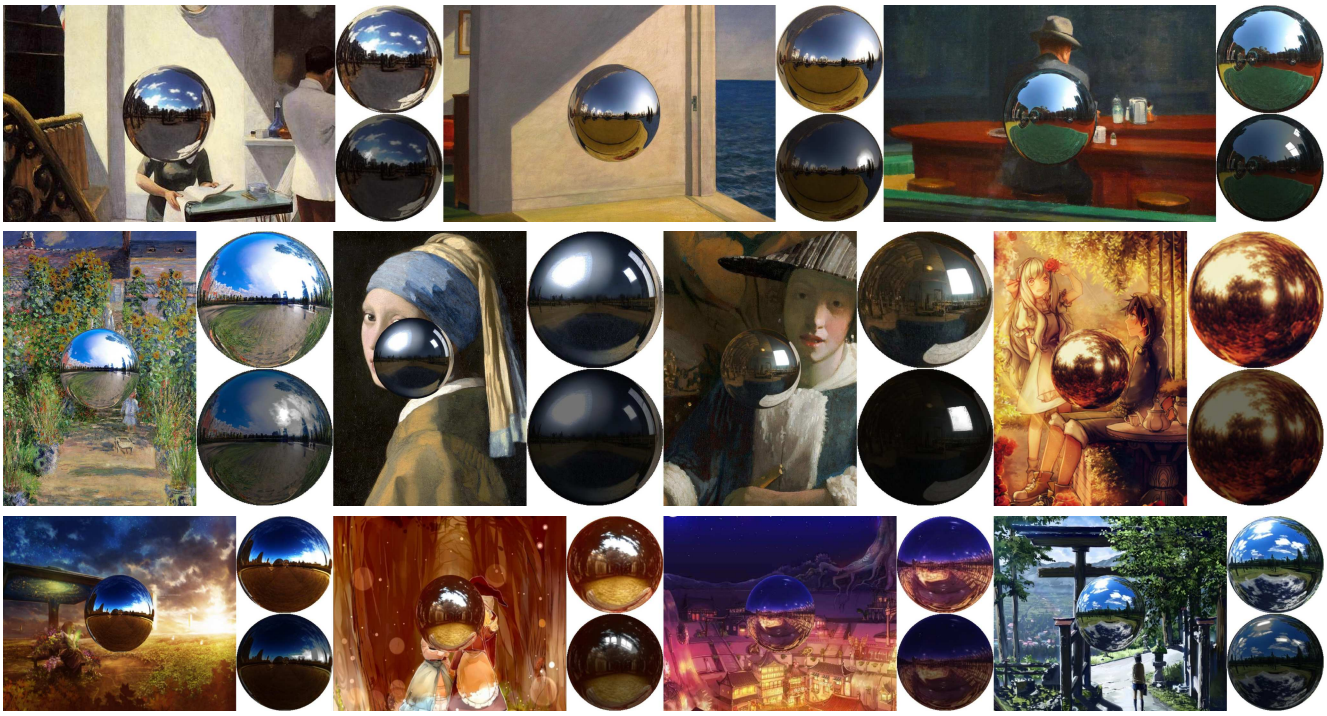


Figure 33. Qualitative results for artificial images such as paintings and painting-like Japanese animation-style images. For each input, we show a chrome ball generated from our pipeline and its underexposed version. Our proposed method can still perform reasonably well, albeit with some performance degradation, by leveraging the strong generative prior of SDXL [45].