Supplemental Material for DreamPBR: Text-driven High-Resolution SVBRDF Generation with Multimodal Guidance

In the supplemental material, we provide additional experimental details as well as results from generation experiments, comparison experiments, and ablation studies, showcasing the powerful capabilities of DreamPBR in the field of creative material generation.

I. Implementation Details

DreamPBR is trained on quadruple Nvidia RTX 3090 GPUs. During the training of material LDM (m-LDM), we employ Adam as our optimizer with a base learning rate of 1.6×10^{-3} and closed learning rate scaling. Starting with the stable-diffusion-v1-5 checkpoint [1] for 9000 epochs, we finetune it for approximately 10 days. For the training of the PBR decoder, we set the base learning rate to 4.5×10^{-6} and enabled scale_lr, taking 4 days total in which the output channels of the decoder are set to 8, with albedo and normal having three channels each, and metallic and roughness being single-channel. For the highlight-aware albedo decoder, we set the base learning rate to 4.5×10^{-6} and enabled scale_lr, taking 2 days total in which the output channels of the decoder were set to 3. We incorporate rendering loss during the training process.

During the training of the rendering-aware super-resolution module, we initially utilized the preset weights from Real-ESRGAN [2] and finetuned four super-resolution modules specifically for albedo, normal, metallic, and roughness textures. These modules are finetuned using the learning rates of 1×10^{-4} and 10000 total iter. Furthermore, we combined the training of all four modules in a model to render the result of each module during training and incorporated rendering loss.

For Pixel Control, we set the learning rate to 1×10^{-5} for training ControlNet [3], which requires about 2 days to complete. For Style Control, we directly utilize the ip-adapter_sd15 checkpoint [4] along with our finetuned checkpoint, as we have observed satisfactory results.

II. Generation Results

We categorize our dataset into ten types manually: Brick (58), Fabric (60), Ground (99), Leather (45), Metal (130), Organic (45), Plastic (40), Tile (75), Wall (69), and Wood (90). we obtain a mount of descriptions of materials in by LLM for each type, which is used to sample materials with DreamPBR. The generated textures are enhanced by the super-resolution module and rendered as shown in Fig. 2.

Besides the consistency of text and images, the diversity of results is quite important for text-driven generative models as well. As demonstrated in Fig. 1, we further sample several textures with the same prompt but different random seeds, DreamPBR succeeds in producing diverse textures that follow the descriptions we specify.

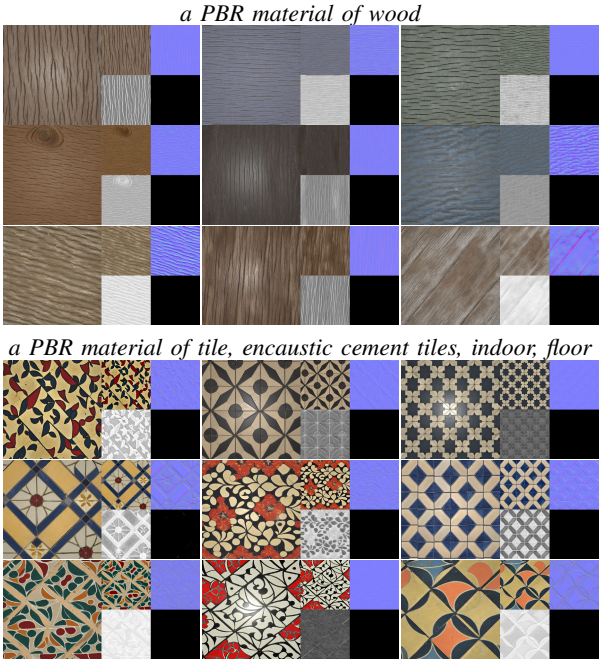

Fig. 1. Diverse sampling results under the same prompts. We evaluate the diversity with the same basic description (top) and the same detailed description (bottom) but different random seeds. Both of them show quite different patterns and textures although the same prompt is used.

Although the users would introduce various controls, we can generate seamless tileable textures all the time, which allows users to apply the generated textures in different scales and different scenes. In Fig. 3, we present several tileable textures from direct and guided generation with their splicing results, showing the effectiveness of circular padding in our method.

With additional control of binary images, inpainting is also a usual method for users to obtain specified results so we present several inpainting results in Fig. 4 to replace a region in texture with another region users describe.

Fig. 5 illustrates the situation in that users would like to combine Style Control with Pixel Control, which enables users to generate the results they want more freely.

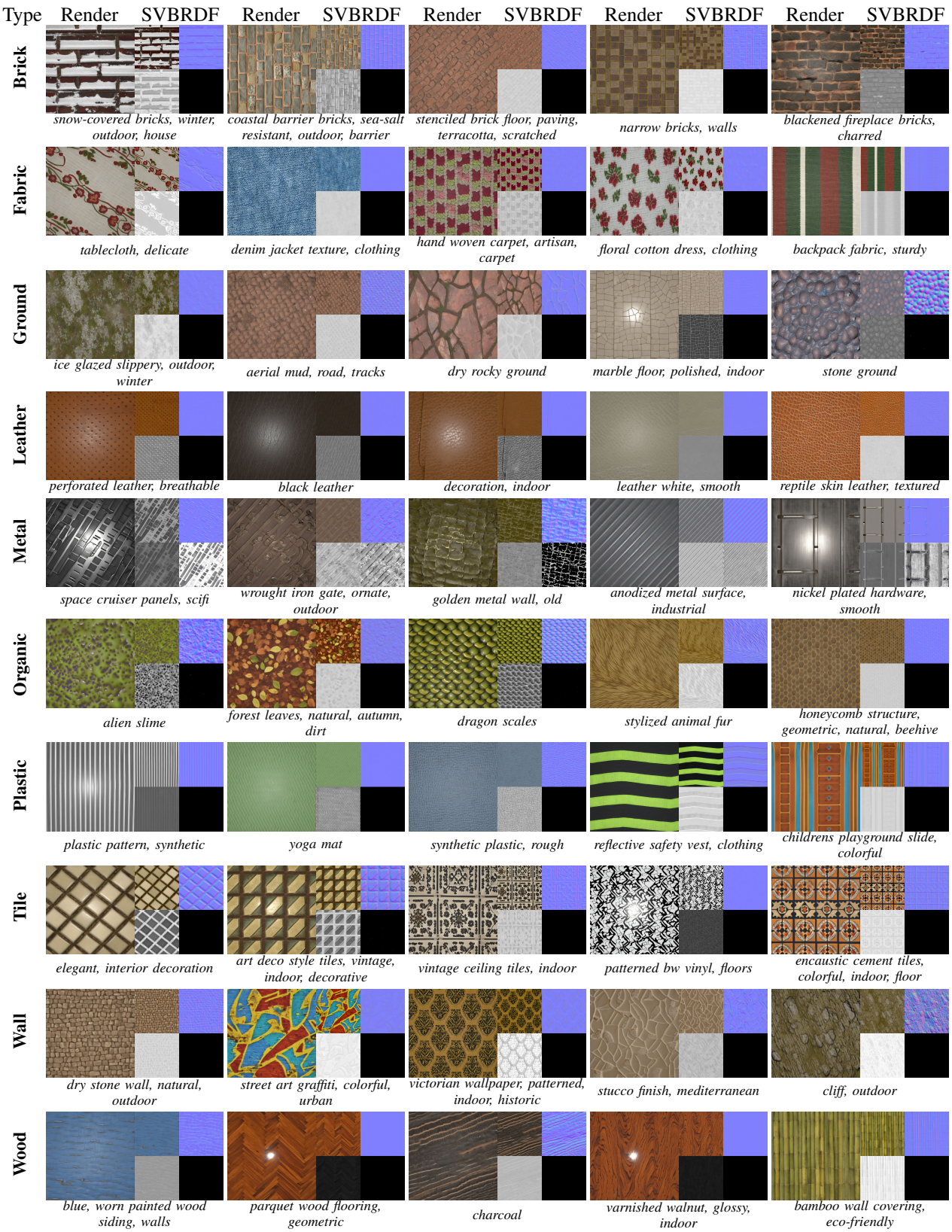

Fig. 2. The generation results of DreamPBR under text-only conditions: We randomly sampled numerous materials with various types and wide tags, by the prompts, "a PBR material of [type], [tags]". Not only can DreamPBR generate materials that match the descriptions, but also some out-of-domain materials are created as well such as **brick** of snow-covered bricks, **plastic** of a children's playground slide, and **wall** of street art graffiti.

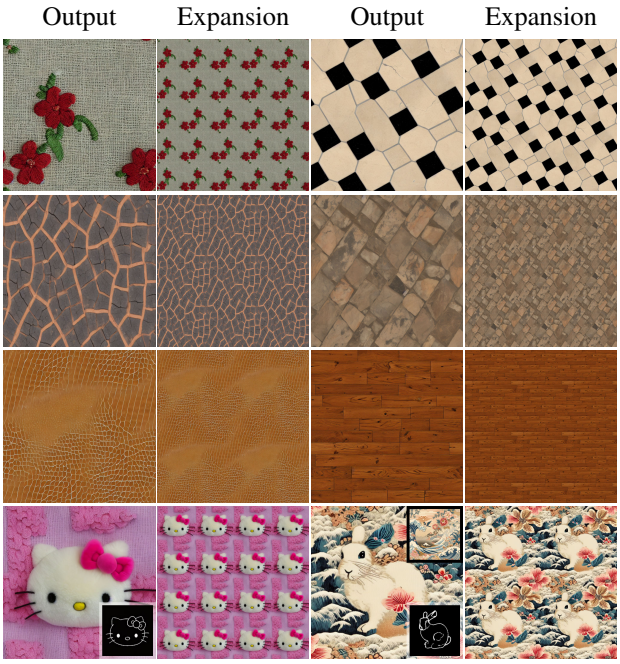

Fig. 3. Splicing results of our tileable textures. The first 3 rows show the tileable results generated under only text descriptions and the last row shows the results with Pixel Control and Style Control, whose control conditions are attached to the edge of images. All the textures we generated in this figure show high tile abilities without artifacts.

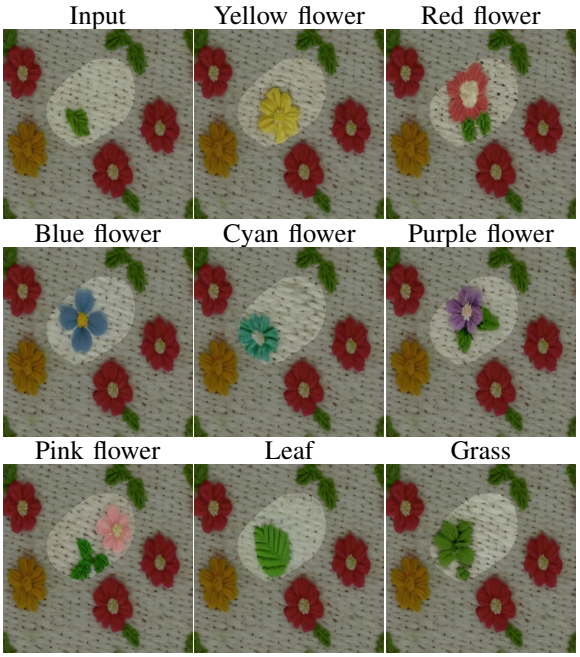

Fig. 4. Inpainting results. The original texture is shown in the upper left corner, which is generated by the prompt "a PBR material of the fabric, floral cotton fabric". By different tags (above each image) and regions to be inpainted (bright areas in each image), we can manipulate user-specified areas in the textures according to preferences such as changing a leaf to colorful flowers.

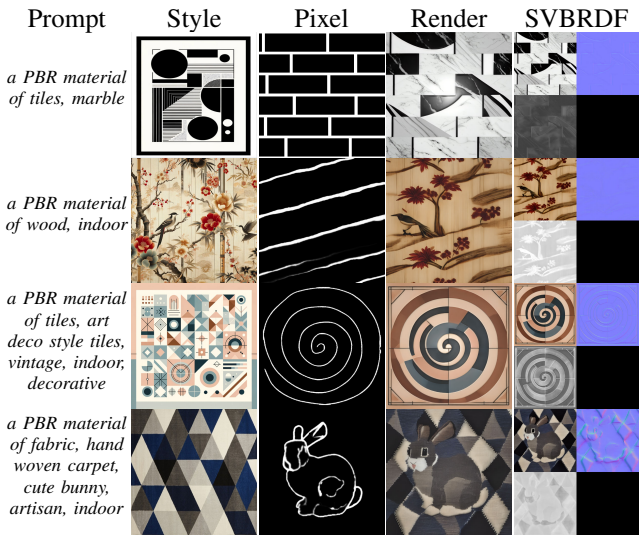

Fig. 5. Results of combined control. We combine descriptions of materials (the first column), styled images (the second column), and binary images (the third column) to control the generated textures. Under the descriptions of materials, the generated results have both the given pattern and style incorporated into them.

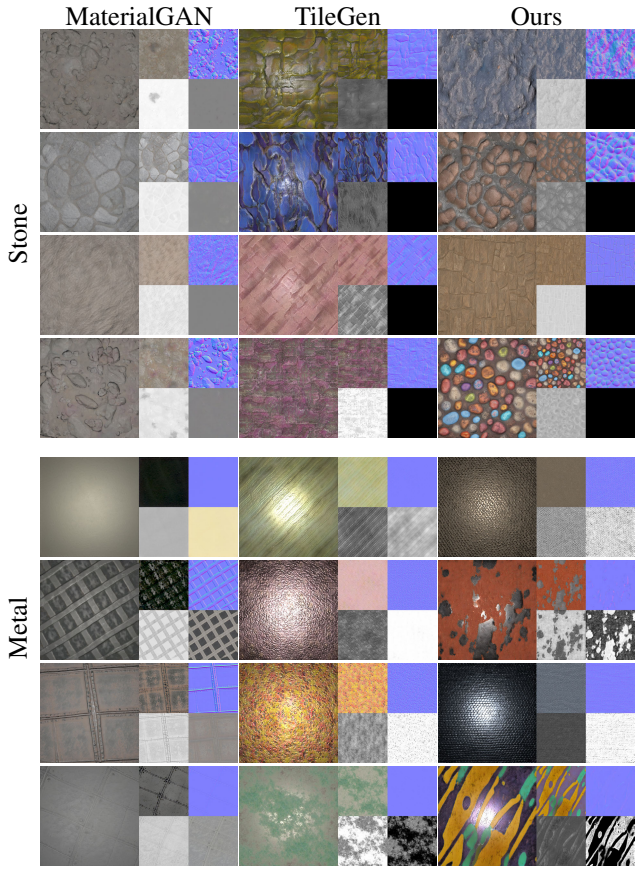

Fig. 6. Qualitative comparison on texture generation. We randomly sample several materials of stone and metal by MarerialGAN and TileGen, which are used to be compared with ours. Importantly, we can generate out-of-domain textures as shown in the fourth row and the last row of ours, which is beyond the capabilities of GAN-based methods.

III. Comparative Experiments

Leveraging the powerful generative model, StableDiffusion, DreamPBR proves previous methods for materials generation. We compare the results generated from DreamPBR for different materials against MaterialGAN [5] and TileGen [6] in Fig. 6. Notably, there are only two categories provided in the competing methods so our results are generated by giving prompts, "a PBR material of ground, stone" and "a PBR material of metal". The comparison shows that DreamPBR can generate textures following the distribution of realistic data from datasets like GAN-based methods as well as magic textures from prior information for 2D images.

IV. Ablation Study

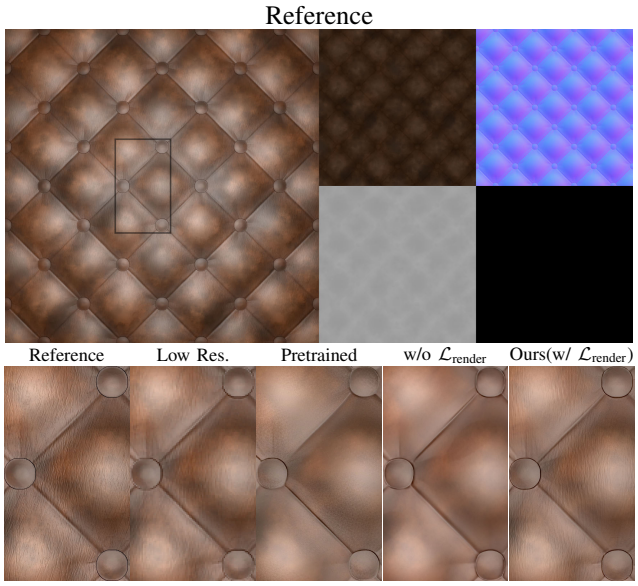

Fig. 7. Qualitative comparisons on the Super-Resolution module are shown, including results with high-resolution textures (2048×2048), low-resolution textures (512×512), and textures from three training strategies: pre-trained, fine-tuned without $\mathcal{L}_{\text{render}}$, and fine-tuned with $\mathcal{L}_{\text{render}}$ (ours). Our method achieves textures and details more consistent with ground truth, as demonstrated in the bottom five images of the figure.

TABLE I
Comparisons on the Super-Resolution Module

Method	LPIPS	RMSE			
	Render	*Albedo*	*Metal.*	*Normal*	*Rough.*
Pretrained	0.450	0.0272	0.0816	0.0598	0.0588
w/o $\mathcal{L}_{\text{render}}$	0.342	0.0248	0.0652	0.0474	0.0451
Ours (w/ $\mathcal{L}_{\text{render}}$)	0.321	0.0211	0.0643	0.0398	0.0445

TABLE II
Comparisons on the Highlight-Aware Module (HA)

Method	Highlight Inputs			Non-Highlight Inputs		
	L1	*PSNR*	*LPIPS*	*L1*	*PSNR*	*LPIPS*
w/o HA	0.0409	25.7460	0.1928	0.0201	33.2621	0.1220
w/ HA	0.0211	32.6578	0.1452	0.0202	33.2904	0.1241

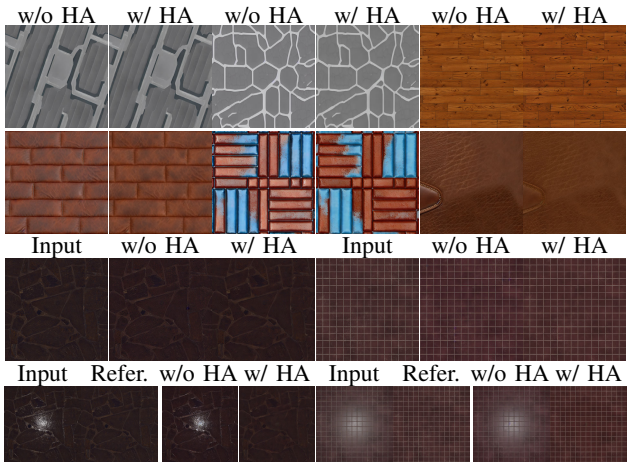

Fig. 8. Qualitative comparisons on the highlight-aware module (HA) show its ability to remove highlights from albedo maps generated by the VAE decoder, improving generation results. For inputs without highlights, the module has minimal impact, while for inputs with highlights, it effectively de-highlights them.

Fig. 9. Qualitative comparison on fine-tuning ControlNet. We evaluate the pre-trained ControlNet and our fine-tuned version based on the pre-trained one to employ pixel-guidance generation. The textures from pre-trained ControlNet (w/o ft) are more like natural images rather than textures.

Although the super-resolution models originally show great results in natural images, we finetune it again with our material data and employ a novel rendering loss $\mathcal{L}_{\text{render}}$ from the level of perception. In practice, we finetune super-resolution modules for each component of textures based on the pre-trained Real-ESRGAN as our baseline. With four single modules(albedo, metallic, normal, and roughness), we jointly finetune them and introduce the $\mathcal{L}_{\text{render}}$ by rendering four textures after super-resolution to image space. The comparison results are shown in Fig. 7. Table. I shows that our final model, fine-tuned with $\mathcal{L}_{\text{render}}$, achieves the lowest LPIPS for rendered images and the lowest RMSE for SVBRDF maps compared to ground truth. Similar to the training of PBR Decoder, the finetuning super-resolution modules with $\mathcal{L}_{\text{render}}$ contributes to better results.

We introduce a highlight-aware albedo decoder to remove the potential highlights in generated RGB images. For a good de-highlight module, there are two key points to be taken into account: (1) effectively removing the highlights in images, and (2) leaving them unchanged for those without highlights. In practice, only training on rendered images potentially affects the decoded albedo(without highlights), so we finetune the highlight-aware decoder by randomly choosing rendered images from different lights or pure albedo maps. Furthermore,

we compare the outputs of the highlight-aware decoder with those of the initially pre-trained decoder in Fig. 8, suggesting that our decoder addresses the issues of two key points above. Table. II confirms this conclusion by comparing L1, PSNR, and LPIPS between real albedo maps and de-highlighted albedo maps.

To realize sketch-guidance control, we embed a pre-trained ControlNet in DreamPBR. However, different from the IP-Adapter for Style Control focuses on incorporating semantics of images in clip space independent of training data, the initial ControlNet leads to domain shift, from the albedo domain back to the image domain, in our experiments. To address this problem, we finetune the ControlNet with our sketch-albedo pairs as mentioned above. The comparison of ControlNet before and after being finetuned is shown in Fig. 9.

REFERENCES

[1] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[2] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan, "Real-esrgan: Training real-world blind super-resolution with pure synthetic data," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 1905–1914.

[3] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala, "Adding conditional control to text-to-image diffusion models," 2023.

[4] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang, "Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models," *arXiv preprint arXiv:2308.06721*, 2023.

[5] Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao, "Materialgan: Reflectance capture using a generative svbrdf model," *ACM Trans. Graph.*, vol. 39, no. 6, nov 2020.

[6] Xilong Zhou, Milos Hasan, Valentin Deschaintre, Paul Guerrero, Kalyan Sunkavalli, and Nima Khademi Kalantari, "Tilegen: Tileable, controllable material generation and capture," in *SIGGRAPH Asia 2022 conference papers*, 2022, pp. 1–9.