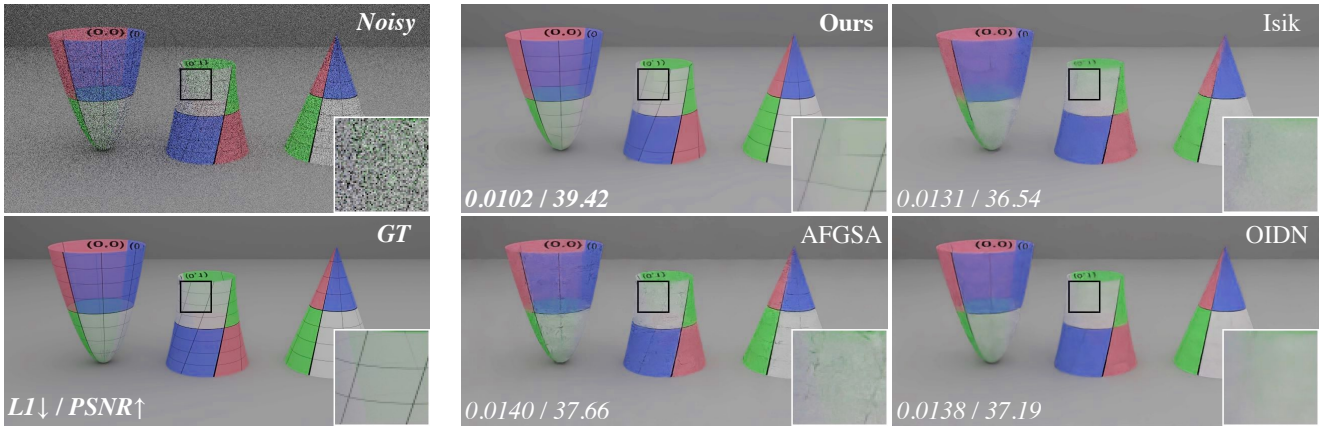
Denoising Monte Carlo Renders with Diffusion Models



We present a denoiser based on a pixel-space diffusion model. Because our method has a strong prior of what a real image looks like, it can generalize better on out of distribution images. In the example above, notice how competing methods produce unwanted artifacts like splotchy/blurry regions and missing/broken-up lines on the surface texture. Our method will consistently produce an image that looks like a real image (e.g. minimal unwanted artifacts), while adhering to the conditioning buffers. Results shown on a 4spp test image.

Vaibhav Vavilala
UIUC

Rahul Vasanth
UIUC

David Forsyth
UIUC

Abstract

Physically-based renderings contain Monte Carlo noise, with variance that increases as the number of rays per pixel decreases. This noise, while zero-mean for good modern renderers, can have heavy tails (most notably, for scenes containing specular or refractive objects). Learned methods for restoring low fidelity renders are highly developed, because suppressing render noise means one can save compute and use fast renders with few rays per pixel. We demonstrate that a diffusion model can denoise low fidelity renders successfully. Furthermore, our method can be conditioned on a variety of natural render information, and this conditioning helps performance. Quantitative experiments show that our method is competitive with SOTA across a range of sampling rates. Qualitative examination of the reconstructions suggests that the image prior applied by a diffusion method strongly favors reconstructions that are "like" real images – so have straight shadow boundaries, curved specularities and no "fireflies."

1. Introduction

The image produced by a physically-based renderer is the value of a random variable. Typically, the mean of this random variable is the (unknown) true result; but light transport effects mean that the variance – the result of using "too few rays" – is complicated and heavy tailed. Analogous effects appear in very low light photography. Increasing N – the number of rays per pixel – quickly results in diminishing returns, because the variance goes down as $1/N$, and so there is a significant literature that aims to suppress render noise. This paper shows that a method based on diffusion is quantitatively competitive with the state of the art (SOTA) while producing images that differ strongly in qualitative aspects.

Obvious solutions to render noise fail. Real-world scenes in the film industry are too large to fit into GPU RAM and must be rendered on a CPU. Often even hundreds to thousands of rays per pixel may fail to reach an artist's desired level of quality. Problems are caused by the presence of desirable but complex light transport phenomena such as indirect specular, large numbers of light sources, subsurface scattering, and volumetric effects.

The concept of relying on pretrained foundation mod-

els has been extensively explored for image restoration [23, 34]. Diffusion models have successfully removed JPEG noise and film grain; images can be recolored; and local objects can be inpainted. Even the success of large language models relies on a foundation model trained for next token prediction, then finetuned for particular tasks (for example, chatbots like ChatGPT relying on GPT-4 as the backbone). **Until now, large foundation models have not been applied to denoising Monte Carlo renders**.

There are multiple reasons for this. For one, the area has matured - high quality denoisers already exist, some geared towards high quality and others towards speed. Another reason is that there are known workarounds when existing denoisers create problems. Artists can sample scenes for longer (requiring additional compute and slowing artist iteration) or manually touch-up unwanted artifacts in finishing software like Photoshop or Nuke.

Further, it is not obvious that image foundation models can handle high dynamic range gracefully, since they are trained on images with per-pixel radiance from $[0-1]$. Ray traced images in contrast store the per-pixel colors in linear space, which may exceed 1 if rendering a bright or shiny object. Subsequent processing such as gamma correction, tonemapping, compositing, and color grading bring the final values to $[0-1]$, suitable for standard dynamic range (SDR) display. Denoisers must process inputs in a manner respecting high dynamic range to be useful in practice. **Across the billions of images foundation models are trained on, a variety of post-processing will be reflected** in the datasets such that the result sits in the range $[0-1]$. We believe that **this wide array of tonemapping, geometry, textures, and lighting is an asset if the foundation model can be utilized effectively.** As we show, images denoised with foundation models can often look better than existing methods.

Another potential pitfall is that large scale training sets will include images with unwanted lossy compression artifacts, film grain, image rescaling, motion blur, and other phenomena associated with the capture and storage of imagery. It's not obvious whether an image can be effectively denoised without unwanted artifacts or hallucinations leaking into the result. In practice, we don't notice this to be a problem in our evaluation.

Expensive data collection is another concern - it is extremely compute-intensive to generate 3D scenes and render them with a path tracer to an acceptable level of quality. A single image may require tens to hundreds of CPU hours to converge, and tens of thousands of frames are required to build a sufficient training set. While high quality rendering software exists (such as RenderMan used in VFX houses, PBRT and MITSUBA for research [11, 28, 30]), high quality datasets are not freely available. The research community would benefit from large open source renders of practical scenes with feature buffers at low sample counts, and accompanying high sample target renders.

A problem we aim to address in this work is that existing denoisers frequently struggle to generalize to scenes with phenomena different from the training set (for example different hair colors/styles, volumetric effects, and novel textures). That may result in users having to collect more training data to retrain an existing denoiser (a computationally demanding effort). Alternatively, denoiser limitations may result in additional artist effort to detect and fix problems manually - or simply accepting a lower-quality product.

Advancing the SOTA for denoising may alleviate some of these problems. This paper is firmly in the high quality regime (as opposed to focusing on speed), as we are utilizing a multi-pass diffusion model and evaluating against the best existing methods, which are all 1-pass neural networks. We however present suggestions for making our model run much faster.

Our contributions are:

1. We are the first to use large-scale image generation foundation models to denoise MC renders, and we demonstrate that conditioning on render buffers provides essential information to the diffusion model. To do so, we apply the ControlNet architecture to a pixel-based (rather than latent variable based) diffusion model.
2. Quantitative and qualitative evidence suggest our method is generally better than existing SOTA methods.

2. Related Work

Early denoisers relied on linear regression models and hand-designed filters that run quickly. Zwicker et. al. provide an excellent survey on classical approaches [51], while Huo et. al survey deep learning methods on denoising Monte Carlo renderings [17]. The latest advancements in MC denoising reflect the progression of learning-based approaches. We focus our literature review on recent denoising works and controlled image synthesis.

2.1. Real-time and Interactive Denoisers

Chaitanya et. al lower temporal noise in animations and are the first to use a U-net [9, 33]. Thomas et. al incorporate a U-net, using a low-precision feature extractor and multiple high-precision filtering stages to perform both supersampling and denoising [39]. Fan et. al make improvements to efficiency, building on the hierarchical approach utilized by Thomas et. al by predicting a kernel for a single channel and incorporate temporal accumulation [13]. Lin et. al employ a path-based approach [24].

Meng et. al train a network to splat samples onto multiscale, hierarchical bilateral grids, then denoise by slicing the grid [26]. Lee et. al incorporate kernel prediction and temporal accumulation for real time denoising [20, 21]. Munkberg et al. build on the approach used by Gharbi et.

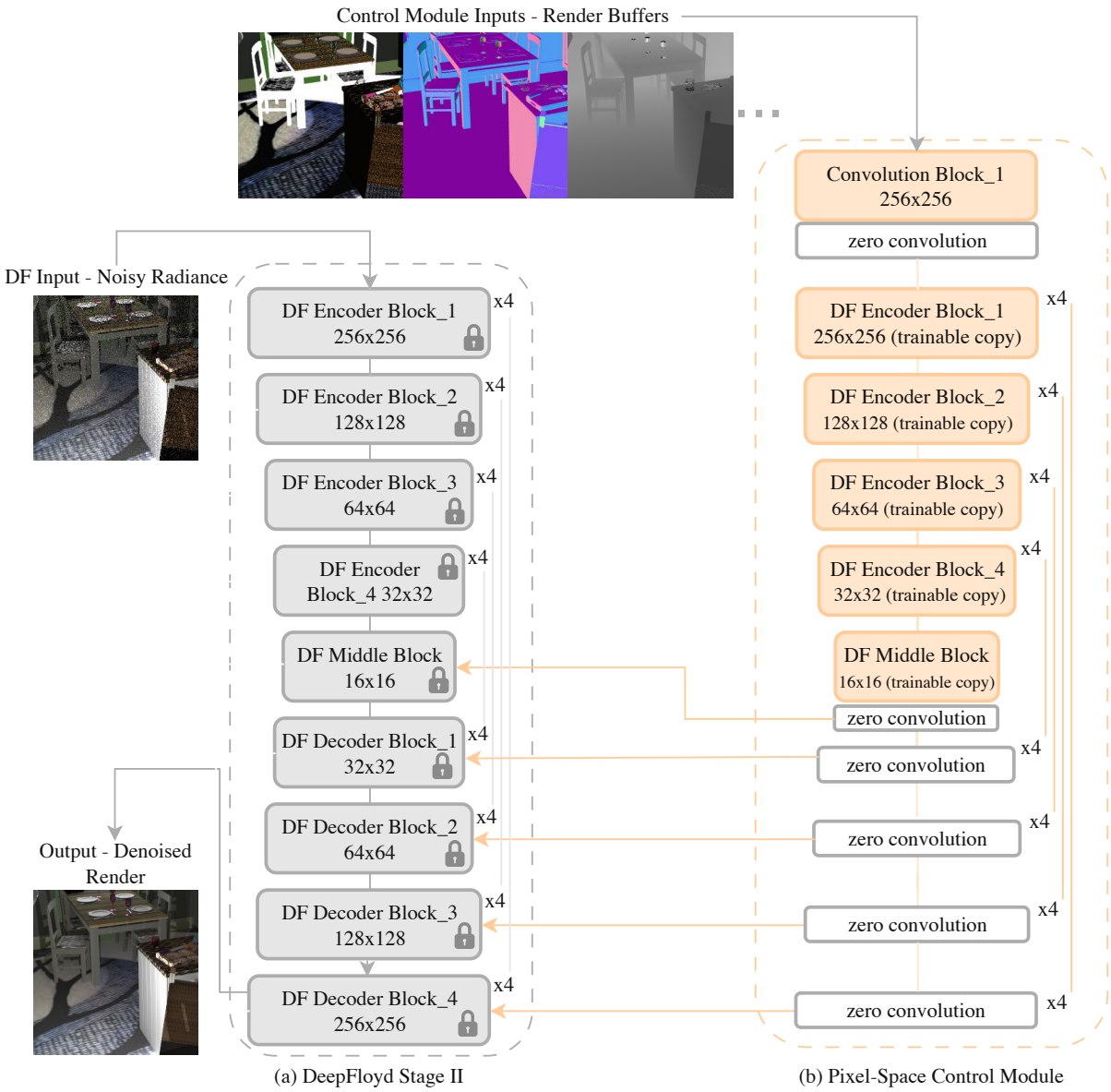


Figure 1. **Overview of our method.** We leverage a pretrained pixel-space diffusion model, DeepFloyd Stage II [2], as our base synthesizer that is fixed during training. It accepts the noisy radiance as well as a forward-diffused copy of the noisy radiance (see methodology section 3 for details). We introduce a trainable Control Module, analogous to ControlNet [50], initialized with the encoder and middle blocks. It accepts all the auxiliary feature buffers from the renderer like albedo, normals, and depth, in addition to the noisy radiance. The outputs of the Control Module are added to the DF decoder blocks at varying spatial resolutions. We utilize zero convolutions to ease early stages of training. Time and prompt encoding not shown for brevity. Our prompt is the empty string during training and inference.

al, but splat radiance and sample embeddings onto multiple layers. Filter kernels are applied to layers which are composited, improving performance [27]. Munkberg et. al eschew Laplacian pyramids used by Vogels et. al and improve temporal stability compared to Chaitanya et. al. by maintaining a per-sample approach [14, 27, 45].

Similar to Munkberg et. al, Isik et. al use per-sample information when computing filter weights, though the filters operate on pixel-wise averages. Their network predicts dense features, which are then incorporated in a pairwise-affinity metric that results in per-pixel dilated 2D blur kernels applied iteratively to the low sample radiance. An optional temporal kernel can blend the previous frame's denoised output with the current frame [18, 27].

Figure 2. Example images from our rendered dataset.

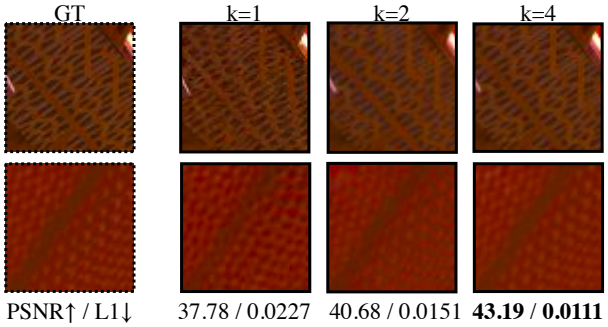


Figure 3. *Best viewed online in color.* Diffusion methods based on a latent variable image representation will not work for render denoising, because the VAE decoding creates significant problems. Existing ControlNet [50] architectures use a latent representation of the image. The limited size of the VAE dictionary limits the accuracy of very precise pixel space tasks. Upsampling the image helps, but does not remove this effect. In each row, 64x64 cropped training images are shown, highlighting texture. We take the GT training image, upsample by k, feed it through a VAE, decode, downsample by k and show the result. We use the default VAE from Stable Diffusion 2.1. Error metrics shown underneath are averaged over 64 random training images. The VAE introduces unacceptable changes to texture. In the first row, the black patches are sharply defined in GT, but blurred after VAE decoding. In the second, the VAE shifts the color of the texture. These changes are partially due to the 8x spatial downsampling factor of VAEs, converting a (H,W,3) image into a (H/8, W/8, 4) dimensional latent code. Thus, existing control mechanisms relying on latent-space diffusion models like ControlNet are not suitable for pixel-space tasks like MC denoising. Even with 4x upscaling, which defeats much of the efficiency gain of the VAE, the PSNR is comparable with existing SOTA denoisers, which effectively caps the quality that can be achieved. Thus, in this work we introduce spatial controls to pixel-space diffusion models.

2.2. Offline Denoisers

Kalantari et al. utilize a multi-layer perceptron at a per-pixel level to optimize cross-bilateral filter parameters [19].

Rather than hand-crafted kernels, Bako et al. use a convolutional neural network to predict filtering kernel weights adaptively at a pixel level, then apply the kernels to the noisy image input. Specular and diffuse components are separately processed. The kernel prediction convolutional network (KPCN) and kernel-prediction approaches are popular as they train quickly and are more robust than predicting colors directly [5].

Vogels et al. utilize hierarchical pyramid kernels, approximating the behavior of large kernels with a small kernel multi-resolution cascaded filtering strategy. This work extends KPCN to animated sequences, incorporating an adaptive sampling approach with temporal aggregation [45]. Instead of inferring weights per pixel, Gharbi et al. use radiance samples, demonstrating their utility for denoising by predicting splatting kernels for each sample [14]. Balint et al. further develop the pyramidal filtering approach with improvements to upsampling, weight predictor networks, and learnable partitioning [6].

Xu et. al show that generative adversarial networks can be used for denoising, eliminating in-between layers used in Bako et. al, but similarly train separate networks to process specular and diffuse components [5, 47]. Yu et. al develop this adversarial approach by processing all components together and incorporate a modified self-attention: auxiliary feature guided self-attention (AFGSA) [49]. This work effectively implements a global blur kernel since each pixel attends to the features of all pixels via the cross-attention mechanism. Back et. al apply a post-processing network incorporating a self-supervised loss to KPCN, AFGSA, and Xu et. al [47] to improve denoising quality [4].

Our work deviates from much of the recent kernel-based approaches in that we predict the colors directly.

2.3. Diffusion Models

Stable Diffusion is a foundational model that implements latent diffusion [15, 32, 38], whereby a VAE compresses the spatial resolution of the input image, the diffusion model is run in latent space, and the resulting latent code is decoded into the output image. The synthesis is commonly conditioned by text input via cross attention. Additional controls can be introduced, most notably ControlNet [50]. ControlNet has been shown to facilitate control over latent diffusion models by conditioning image synthesis on arbitrary spatial information like edges, depth, and segmentation. Additional forms of conditioning include color palettes [44] and even 3D primitives [40, 42, 43]

Imagen is another successful large-scale image synthesis model that operates in pixel space [35], with image quality comparable to Stable Diffusion. Imagen relies on three diffusion models: the first synthesizes at 64x64 resolution; the second upsamples to 256 resolution; and the third upsamples to 1024. Thus, super resolution replaces the need for a VAE, easing compute requirements. DeepFloyd is an open source implementation of Imagen [2]. We remark that similar to denoising, super resolution is highly practical in real-world rendering systems [41].

Wang et. al pass pixel-level features to a pretrained Stable Diffusion model for restoration [46]. Yang et. al introduce a pixel-aware cross attention module to latent diffusion models for realistic image super-resolution [48]. Similarly, Instruct-Imagen utilizes cross-attention to condition a pixel space diffusion model on spatial layout [16]. In contrast, we use concatenation and addition to condition our diffusion model, like ControlNet, but in pixel space.

3. Method Overview

3.1. Data

Rendering We generate approx. 5650 random scenes using a procedure similar to [18]. Each scene contains randomly arranged ShapeNet objects, textures, and lighting configurations [10]. For each scene, we create a random camera move over the course of 8 frames and render with PBRT-V3 at 256 resolution [30]. We render with the path-tracing integrator and allow several bounces to capture indirect illumination. For each frame, we render the ground truth GT^1_{raw} at 4096 spp, and render 16 additional frames with feature buffers at 4spp (all with different random seeds to avoid noise correlations). Thus we can train our denoiser with spp $\in \{4, 16, 64\}$. Our auxiliary feature buffers F include normals, albedo, depth, direct specular, indirect specular, direct diffuse, indirect diffuse, roughness, emissive, metallic, and transmission. Variance for each channel is also stored.

Data sanitization Unfortunately, our target renders were still a bit noisy at such a high spp due to indirect illumination noise. Noise2Noise [22] establishes that neural networks can learn to denoise even when ground truth is noisy if the training objective (often an L1 or L2 loss) attempts to recover the mean of the distribution. This assumption allows several prior methods to succeed even when ground truth is noisy. However, methods relying on a discriminator such as AFGSA [49] are not compatible with noisy ground truth because noise will leak into the generated distribution. Thus we adopt the following procedure to post-process the ground truth images. We render an additional GT image GT^2_{raw} at 4096 spp for each scene, but use a different random seed than GT^1_{raw}. We then train a network with only SMAPE loss that accepts a GT render and predicts the other one from the pair that we rendered. This data-cleaning network C_θ is optimized with:

$$L_{\text{SMAPE}} = \frac{1}{N} \sum_i \text{SMAPE}(C_\theta(F_i, GT^1_{i,\text{raw}}), GT^2_{i,\text{raw}}) \quad (1)$$

Where the loss is averaged over $i = 1, 2, ...N$ samples in a minibatch and $GT^1_{i,\text{raw}}$ and $GT^2_{i,\text{raw}}$ are interchanged at random during training. SMAPE is defined as:

$$\text{SMAPE}(A, B) = \frac{||A - B||_1}{||A||_1 + ||B||_1 + \epsilon} \quad (2)$$

We adopt the architecture of Isik [18] for C_θ, conditioning the input on features but excluding any temporal loss terms. The fully-trained data cleaning network is then run on all the ground truth images we rendered (we just pick one out of each pair of raw GT frames), generating clean GT renders:

$$GT_{i,\text{clean}} = C_\theta(F_i, GT^1_{i,\text{raw}}). \quad (3)$$

All evaluated methods use GT_{clean} as the target during training, and have access to the same training, validation, and test splits.

Range-compression Large-scale pretrained diffusion models are trained on images with various degradations and tonemappings applied, ultimately with pixel values ranging from 0 to 1. Our dataset however has useful radiance values well beyond that - we clamp the radiance buffers to 6. The practice of range-compressing in log space is common across all MC denoising works we evaluated. In our case, we observe that gamma-correcting tonemappers produce more natural-looking results consistent with datasets diffusion models were trained on. Thus we conduct initial experiments with the following tonemapper:

$$V_{\text{out}} = AV_{\text{in}}^{\gamma} \quad (4)$$

We set $A = 0.47, \gamma = 1/2.4$, fully capturing the $0 - 6$ range. However, early results showed that log tonemapping produces better results for our model:

$$V_{\text{out}} = A * log(1 + V_{\text{in}}) \quad (5)$$

We set $A = 0.51$. In our evaluation, we show that our diffusion model can effectively denoise images in a LDR space, and we reverse-tonemap the synthesized result back to HDR space for error metric calculation. All figures in this manuscript are tonemapped with gamma (eqn. 4), as the details appear more clear.

Additional test data We hold out 3% of the data for validation and 5% for testing. For qualitative evaluation, we render a few PBRT-v3 test scenes that portray practical real-world scenarios[1] though we do not apply post-processing to the ground truth as we did for the training images.

[1] https://www.pbrt.org/scenes-v3

For 4 spp models, we randomly sample one low-res stack of buffers out of 16 available; we sample 4 buffers for the 16 spp models; and use them all for the 64 spp models. The aux buffers are averaged over the selections.

3.2. Architecture

We use DeepFloyd [2] (which is based on Google's Imagen [35]) as our base architecture, and build a control mechanism around it in a manner similar to ControlNet [50]. Our Control Module, shown in Fig. 1, consists of a few convolutional layers that accept arbitrary numbers of channels (39 aux buffers in our case), the DeepFloyd encoder layers, and the DeepFloyd middle layer. The Control Module generates feature maps at resolutions (256, 128,...16) which are summed with the original DeepFloyd encoder outputs and passed to the DeepFloyd decoder. DeepFloyd modules are not trainable, only the Control Module is, consistent with the original ControlNet. Zero convolutions are initialized to ensure the system produces realistic images from the very beginning. We only use the DeepFloyd Stage II module for this work, which is intended to perform super resolution; we show it can perform image denoising here. We use the smaller IF-II-M model for efficiency, which has 450M parameters. Our base module accepts two 3 channel inputs that are concatenated: a time-dependent noisy radiance buffer $q(\mathbf{z}_t \mid \mathbf{x})$, which undergoes the diffusion forward noising process, and that same noisy radiance x without forward noising (because the aug_level parameter is fixed to 0 during training and inference). The Control Module accepts the feature buffers F_i. The forward diffusion process is defined as:

$$q(\mathbf{z}_t \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I}\right),$$
$$q(\mathbf{z}_t \mid \mathbf{z}_s) = \mathcal{N}\left(\mathbf{z}_t; (\alpha_t/\alpha_s)\, \mathbf{z}_s, \sigma_{t|s}^2 \mathbf{I}\right) \quad (6)$$

where $0 \leq s < t \leq 1, \sigma_{t|s}^2 = \left(1 - e^{\lambda_t - \lambda_s}\right)\sigma_t^2$, and α_t, σ_t define a differentiable noise schedule whose log signal-to-noise-ratio, i.e., $\lambda_t = \log\left[\alpha_t^2/\sigma_t^2\right]$, decreases with t until $q(\mathbf{z}_1) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. For generation, the diffusion model is trained to reverse this forward process. We refer the reader to [35] for additional details.

Losses We use the usual losses documented in [2, 35], i.e., mean squared error and variational lower bound. We let the text prompt be the empty string during training and inference.

Training & Inference We train with AdamW optimizer, mixed precision, and batch size 12. While the model is trained at 256 res, it can run inference at varying spatial resolutions as it relies on convolutional and transformer layers. We use the SUPER27 DDPM inference schedule. We experimentally set the aug_level parameter to 0 during training and inference. This parameter is intended to augment the Stage II input, which typically comes from Stage I. In our case we supply the noisy radiance $\mathbf{x}$ (which is already at the target resolution). We train for 7 days (170 epochs) with initial learning rate 0.00002 and halved every two days. All experiments are conducted on one NVIDIA A40 GPU. We train an independent network for each sampling rate (4/16/64) though we would expect one appropriately-trained network to succeed on all spp's in practice. We train independent models for competing methods as well.

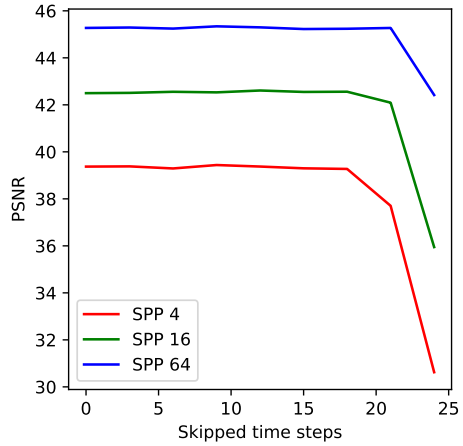


Figure 4. Our method requires approx. 2.8 seconds to denoise a 256x256 image and 63 seconds to denoise an HD 1080x1920 frame (without skipping time steps) on an A40 GPU. We use mixed-precision during inference and SUPER27 DDPM sampler in DeepFloyd. This experiment suggests that we can skip around 18 of the 27 denoising time steps with negligible loss in quality, making diffusion models even more practical. At inference time, we can add gaussian noise to the low-spp render via equation 6 instead of starting from pure noise. The gaussian noise strength needs to be sufficient to overcome the variance in the noisy render. Even though diffusion is more expensive than single-pass methods, the cost of denoising remains much smaller than the cost of rendering real-world scenes to convergence, which can be dozens of hours. Even scenes that fit into GPU memory can take dozens of minutes to render to convergence, which dwarfs the cost of denoising low-ray estimates.

4. Evaluation

We focus our comparative evaluation on 3 prior methods known for quality - Isik, AFGSA, and OIDN [1, 18, 49]. Isik and OIDN rely on CNN U-nets, and AFGSA uses a transformer with adversarial loss. We evaluate the best available OIDN pretrained model (OIDN-VERSION = 2.1.0) and supply it at test time with noisy radiance, albedo, and normals. We retrain Isik and AFGSA with default hyperparameters, all feature buffers, and best-performing model on the held-out validation set used for testing.

We evaluate each method using two standard metrics: L1

Table 1. **Quantitatively, our method is competitive with SOTA**. We compare error metrics across different spp settings (4, 16, 64) for various methods on our test set, consisting of 226 sequences of 8 frames each at 256 res. All methods are run in single-frame mode and evaluation metrics do not take into account temporal performance. Our method outcompetes all others in L1, which is computed in HDR $[0-6]$ space, as well as FoVVDP. We are also competitive in other metrics, which evaluate both pixel-wise and perceptual quality. DINO [12, 29] and CLIP measure cosine similarity of the generated and GT global image feature vector. In the **fourth row**, we disable our Control Module and denoise with an off-the-shelf DeepFloyd Stage II model, presenting the best numbers after ablating the number of skipped denoising time steps and the aug_level parameter. Applying pure DeepFloyd to low-ray estimates is not successful. **Fifth row** Thus, conditioning our method with render buffer information via our Control Module makes a significant difference to performance.

Method	**L1** ↓	**PSNR** ↑	**LPIPS** ↓	**DINO** ↑ [29]	**CLIP** ↑ [31]	**FliP** ↓ [3]	**FoVVDP** ↑ [25]
AFGSA [49]	0.0279	38.672	0.1130	0.914	0.932	0.0494	8.699
Isik [18]	0.0499	38.828	0.0871	0.945	0.955	**0.0476**	8.835
OIDN [1]	0.0638	36.329	0.1192	0.915	0.904	0.0691	8.645
DeepFloyd-II [2]	0.1009	27.583	0.3860	0.742	0.804	0.1390	6.513
Ours	**0.0237**	**39.130**	**0.0748**	**0.948**	**0.965**	0.0487	**8.888**

4spp

Method	**L1** ↓	**PSNR** ↑	**LPIPS** ↓	**DINO** ↑ [29]	**CLIP** ↑ [31]	**FliP** ↓ [3]	**FoVVDP** ↑ [25]
AFGSA [49]	0.0184	42.076	0.0708	0.939	0.958	0.0350	9.222
Isik [18]	0.0449	42.044	0.0560	0.964	0.972	0.0346	9.283
OIDN [1]	0.0537	39.869	0.0858	0.937	0.924	0.0459	9.158
DeepFloyd-II [2]	0.0825	28.988	0.350	0.775	0.819	0.1076	7.150
Ours	**0.0156**	**42.343**	**0.0499**	**0.965**	**0.975**	**0.0338**	**9.328**

16spp

Method	**L1** ↓	**PSNR** ↑	**LPIPS** ↓	**DINO** ↑ [29]	**CLIP** ↑ [31]	**FliP** ↓ [3]	**FoVVDP** ↑ [25]
AFGSA [49]	0.0142	**45.091**	0.0467	0.956	0.972	**0.0260**	9.562
Isik [18]	0.0433	45.055	**0.0341**	**0.977**	**0.982**	**0.0260**	9.593
OIDN [1]	0.0488	42.664	0.0640	0.954	0.940	0.0329	9.482
DeepFloyd-II [2]	0.0696	29.777	0.317	0.786	0.830	0.0973	7.261
Ours	**0.0113**	44.953	0.0346	0.975	**0.982**	0.0261	**9.616**

64spp

(applied in HDR space) and PSNR (in LDR space). We also evaluate several perceptual metrics: DINO and CLIP feature similarity, LPIPS, FoVVDP, and FliP. For PSNR and perceptual metrics, we tonemap GT and prediction via the following tonemapper, consistent with existing evaluation methods:

$$V_{out} = \frac{V_{in}}{1 + V_{in}} \tag{7}$$

Quantitative evaluation We test each method on 226 sequences of 8 frames each at 256 res in Table 1. Across spp $\in \{4, 16, 64\}$, our method is the top-performer as measured by L1 and FoVVDP. We are competitive with other methods as measured by PSNR and several perceptual metrics. For our method, we use 27 DDPM steps at mixed-precision. However, in Fig. 4, we ablate a key test-time hyperparameter. We show that we can skip over half the DDPM steps with negligible loss in PSNR, resulting in additional savings. This behavior is consistent across all spp's we tested.

Finally, in the fourth row of each table in Tab. 1, we examine the effects of removing the Control Module and denoising with an off-the-shelf DeepFloyd Stage II model, finding that the results are much worse. We test several skipped time step and aug_level values, and report the best numbers. We skip 6 time steps and set $aug_level = 0.5$ in this row. Removing the Control Module hurts quality, thus we conclude it is necessary to obtain good results.

Qualitative evaluation While our error metrics are generally better, the value of pretrained diffusion models becomes clear in our qualitative evaluation. The key finding from our method is that images look more plausible and realistic because the image model has seen billions of images. We demonstrate this in Fig. 5. In some cases, competing methods produce numbers comparable or slightly better than ours. But upon closer inspection, the details rendered from our diffusion model look more realistic, because our model has a strong prior of what a real image looks like. Thus shadows, specular highlights, and undersampled edges look reasonable. Other methods overblur undersampled regions, fail to remove fireflies even when trained to

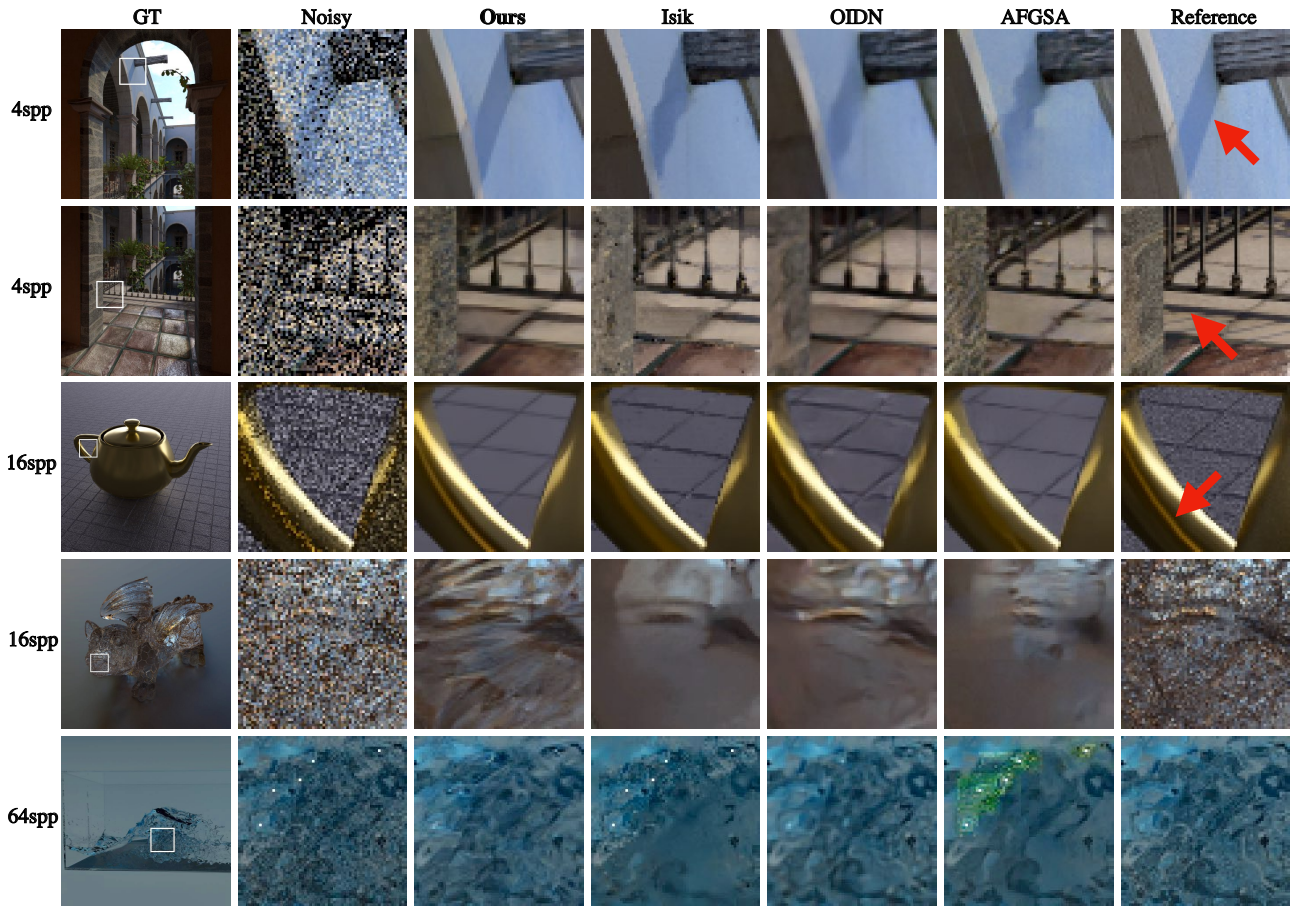


Figure 5. Qualitatively, our reconstructions look like real images, because DeepFloyd has a very strong notion of what an image looks like (it has seen a huge dataset) and because the conditioning buffers offer strong guidelines (e.g. normals, albedo, and depth). In the **final column**, red arrows point to areas of interest. **First row.** Notice the straight edge on the shadow (ours; real images tend to have straight shadow edges) compared with blurred or blotchy edges (others). **Second row.** In undersampled regions, our method fills in the shadow underneath a railing (real images do not have incomplete shadows); other methods render a blurred or incomplete shadow. **Third row.** Notice the clean sharp highlight on the teacup handle, and smooth highlight boundaries (ours; real images have clean sharp highlight boundaries) compared with absent highlights and blotchy boundaries (others). Notice also aliasing effects in the background, most prominent for OIDN, and absent from our reconstruction. **Fourth row.** All methods have problems with this specular dragon. Competing methods overblur the dragon's mouth, whereas ours hallucinates plausible details. **Fifth row.** Fireflies, also known as spike noise, are single very bright pixels, which do not occur in real images. It is rare that fireflies appear at 64 spp in our training set, so AFGSA and Isik fail to remove them. OIDN succeeds in removing them, likely because we use their pretrained model that has seen a large dataset.

do so, and hallucinate splotchy patches when pushed outside the training distribution. Our method is not explicitly engineered to remove fireflies or hallucinate nice-looking specularities and shadows; it only knows it should synthesize a realistic image that follows the conditioning.

5. Conclusion

Pretrained large-scale image models enable MC denoising that is quantitatively competitive with SOTA and qualitatively more realistic. These models that have seen billions of images are clearly beneficial for the denoising task, where expensive, curated training sets typically number in the tens to hundreds of thousands of scenes. One-step models like GANs and distilled diffusion models may yield significant efficiency gains with acceptable quality tradeoffs [36, 37].

One area of future work is what conditioning to use; another is what effects to capture in a training set. Our training set is relatively straightforward, though we test on images with challenging effects. A diverse training set that aggressively oversamples difficult effects might produce better results. Another area of future work is video. As video generators become available [7, 8], our approach might usefully be extended to temporally-coherent denoising.

References

[1] Attila T. Áfra. Intel® Open Image Denoise, 2024. https://www.openimagedenoise.org. 6, 7, 1, 2

[2] Stability AI. Deepfloyd if. 3, 5, 6, 7

[3] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):15:1–15:23, 2020. 7, 2

[4] Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. Self-supervised post-correction for monte carlo denoising. In *ACM SIGGRAPH 2022 Conference Proceedings*, New York, NY, USA, 2022. Association for Computing Machinery. 4

[5] Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2017)*, 36(4), 2017. 4

[6] Martin Balint, Krzysztof Wolski, Karol Myszkowski, Hans-Peter Seidel, and Rafał Mantiuk. Neural partitioning pyramids for denoising monte carlo renderings. In *ACM SIGGRAPH 2023 Conference Proceedings*, New York, NY, USA, 2023. Association for Computing Machinery. 4

[7] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets, 2023. 8

[8] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. 8

[9] Chakravarty Reddy Alla Chaitanya, Anton Kaplanyan, Christoph Schied, Marco Salvi, Aaron E. Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36:1 – 12, 2017. 2

[10] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[11] Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, et al. Renderman: An advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics (TOG)*, 37 (3):1–21, 2018. 2

[12] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2023. 7, 2

[13] Hangming Fan, Rui Wang, Yuchi Huo, and Hujun Bao. Real-time monte carlo denoising with weight sharing kernel prediction network. *Computer Graphics Forum*, 40(4):15–27, 2021. 2

[14] Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. Sample-based monte carlo denoising using a kernel-splatting network. *ACM Trans. Graph.*, 38 (4), 2019. 3, 4

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 4

[16] Hexiang Hu, Kelvin CK Chan, Yu-Chuan Su, Wenhu Chen, Yandong Li, Kihyuk Sohn, Yang Zhao, Xue Ben, Boqing Gong, William Cohen, et al. Instruct-imagen: Image generation with multi-modal instruction. *arXiv preprint arXiv:2401.01952*, 2024. 5

[17] Yuchi Huo and Sung eui Yoon. A survey on deep learning-based monte carlo denoising, 2021. 2

[18] Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. Interactive monte carlo denoising using affinity of neural features. *ACM Trans. Graph.*, 40(4), 2021. 3, 5, 6, 7

[19] Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. A machine learning approach for filtering monte carlo noise. *ACM Trans. Graph.*, 34(4), 2015. 4

[20] Junmin Lee, Seunghyun Lee, Min Yoon, and Byung Cheol Song. Real-time monte carlo denoising with adaptive fusion network. *IEEE Access*, 12:29154–29165, 2024. 2

[21] Junmin Lee, Seunghyun Lee, Min Yoon, and Byung Cheol Song. Real-time monte carlo denoising with adaptive fusion network. *IEEE Access*, 12:29154–29165, 2024. 2

[22] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data, 2018. 5

[23] Xin Li, Yulin Ren, Xin Jin, Cuiling Lan, Xingrui Wang, Wenjun Zeng, Xinchao Wang, and Zhibo Chen. Diffusion models for image restoration and enhancement–a comprehensive survey. *arXiv preprint arXiv:2308.09388*, 2023. 2

[24] Weiheng Lin, Beibei Wang, Jian Yang, Lu Wang, and Ling-Qi Yan. Path-based monte carlo denoising using a three-scale neural network. *Computer Graphics Forum*, 40, 2020. 2

[25] Rafał K. Mantiuk, Gyorgy Denes, Alexandre Chapiro, Anton Kaplanyan, Gizem Rufo, Romain Bachy, Trisha Lian, and Anjul Patney. Fovvideovdp: a visible difference predictor for wide field-of-view video. *ACM Trans. Graph.*, 40(4), 2021. 7, 2

[26] Xiaoxu Meng, Quan Zheng, Amitabh Varshney, Gurprit Singh, and Matthias Zwicker. Real-time monte carlo denoising with the neural bilateral grid. In *Eurographics Symposium on Rendering*, 2020. 2

[27] Jacob Munkberg and Jon Hasselgren. Neural denoising with layer embeddings. *Computer Graphics Forum*, 39(4):1–12, 2020. 3

[28] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):1–17, 2019. 2

[29] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez,

Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 7, 2

[30] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. MIT Press, 2023. 2, 5

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 7, 2

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 4

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 2

[34] Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models, 2022. 2

[35] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 5, 6

[36] Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. In *International conference on machine learning*, pages 30105–30118. PMLR, 2023. 8

[37] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. *arXiv preprint arXiv:2403.12015*, 2024. 8

[38] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020. 4

[39] Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sung ye Kim, Karthikeyan Vaidyanathan, and Angus Graeme Forbes. Temporally stable real-time joint neural denoising and supersampling. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5:1 – 22, 2022. 2

[40] Vaibhav Vavilala and David Forsyth. Convex decomposition of indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9176–9186, 2023. 4

[41] Vaibhav Vavilala and Mark Meyer. Deep learned super resolution for feature film production. In *ACM SIGGRAPH 2020 Talks*, New York, NY, USA, 2020. Association for Computing Machinery. 5

[42] Vaibhav Vavilala, Seemandhar Jain, Rahul Vasanth, Anand Bhattad, and David Forsyth. Blocks2world: Controlling realistic scenes with editable primitives, 2023. 4

[43] Vaibhav Vavilala, Florian Kluger, Seemandhar Jain, Bodo Rosenhahn, and David Forsyth. Improved convex decomposition with ensembling and boolean primitives, 2024. 4

[44] Vaibhav Vavilala, Faaris Shaik, and David Forsyth. Dequantization and color transfer with diffusion models, 2025. 4

[45] Thijs Vogels, Fabrice Rousselle, Brian Mcwilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. Denoising with kernel prediction and asymmetric loss functions. *ACM Trans. Graph.*, 37(4), 2018. 3, 4

[46] Jianyi Wang, Zongsheng Yue, Shangchen Zhou, Kelvin CK Chan, and Chen Change Loy. Exploiting diffusion prior for real-world image super-resolution. In *arXiv preprint arXiv:2305.07015*, 2023. 5

[47] Bing Xu, Junfei Zhang, Rui Wang, Kun Xu, Yong-Liang Yang, Chuan Li, and Rui Tang. Adversarial monte carlo denoising with conditioned auxiliary feature modulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)*, 38(6):224:1–224:12, 2019. 4

[48] Tao Yang, Xuansong Xie Peiran Ren, and Lei Zhang. Pixel-aware stable diffusion for realistic image super-resolution and personalized stylization. In *arXiv:2308.14469*, 2023. 5

[49] Jiaqi Yu, Yongwei Nie, Chengjiang Long, Wenju Xu, Qing Zhang, and Guiqing Li. Monte carlo denoising via auxiliary feature guided self-attention. *ACM Trans. Graph.*, 40 (6), 2021. 4, 5, 6, 7, 1, 2

[50] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 3, 4, 6

[51] M. Zwicker, W. Jarosz, J. Lehtinen, B. Moon, R. Ramamoorthi, F. Rousselle, P. Sen, C. Soler, and S.-E. Yoon. Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Computer Graphics Forum*, 34(2):667–681, 2015. 2