Supplemental Material
Generative Novel View Synthesis with 3D-Aware Diffusion Models

Eric R. Chan *†[1,2], Koki Nagano*[2], Matthew A. Chan*[2], Alexander W. Bergman*[1], Jeong Joon Park*[1], Axel Levy[1], Miika Aittala[2], Shalini De Mello[2], Tero Karras[2], and Gordon Wetzstein[1]

[1]Stanford University [2]NVIDIA

In this supplement, we first provide additional experiments (Sec. A1). We follow with details of our implementation (Sec. A2), including further descriptions of the model architecture and training process, as well as hyperparameters. We then discuss experimental details (Sec. A3). Lastly, we consider artifacts and limitations (Sec. A4) that may be targets for future work. We encourage readers to view the accompanying supplemental videos, which contain additional visual results.

A1. Additional experiments & ablations

Figure A1: New views generated from out-of-distribution poses. Extreme zooms and large translations may lead to unrealistic views.

A1.1. Extrapolation to unseen camera poses.

In the ShapeNet dataset, cameras are located on a sphere, point towards the centers of the objects and have the same "up" direction during training. We investigate the results of our method when querying out-of-distribution poses at test time in Fig. A1. From a fixed pose, we generate a zoom, a one-dimensional translation of the camera, and a camera roll. Although novel views deteriorate with large deviations from the training pose distribution, the 3D prior present in our method can reasonably tolerate small extrapolations.

A1.2. Percentile results based on LPIPS

Fig. A2 shows our synthesized results on ShapeNet ordered by the percentile of the LPIPS [31] score, with examples that scored best according to the metric at the top and examples that scored worst at the bottom. We compute predictions for the same input and output views across the entire test set. To reduce the effects of randomness, we evaluate 9 realizations for each input, and use only the median image/score when ordering our results. Our method produces consistently sharp outputs (even at the 10^{th} percentile) and maintains overall textures and shapes from the input image.

A1.3. Handling multiple input images

Fig. A3 shows our generated novel views when more than one image is given as the input conditioning information. When only 1 view is given from the back side of the car, the model has the freedom to choose multiple plausible completions for the unseen front side of the car, leading to a high standard deviation (high uncertainty). Adding 2 or 3 views reduces uncertainty (low standard deviation), and the model generates a novel view that is compatible with multiple input views.

A1.4. Effect of distance to input view

As Fig. A4 demonstrates, nearby views provide more valuable information than distant views, thus reducing variance in the output rendering. Consequently, by conditioning autoregressively on nearby views, we narrow the conditional distribution of possible outputs, improving geometric consistency compared to

*Equal contribution.
†Work was done during an internship at NVIDIA.

Figure A2: Our synthesized novel views sorted by the percentile of the LPIPS [31] score, with results that scored best according to LPIPS at the top.

non-autoregressive conditioning.

A1.5. Classifier-free guidance

Recently, [9] suggested classifier-free diffusion guidance technique to effectively trade off diversity and sample quality. At training, we implement classifier-free guidance by dropping out the feature image with 10% probability; in its place, we replace this conditioning image with a sample of Gaussian noise. At inference, we can linearly interpolate between unconditional and unconditional predictions of the denoised image in order to boost or decrease the effect of the conditioning information.

Fig. A5 shows the effect of classifier-free guidance [9] (CFG) when making predictions in isolation. In general, positive classifier-free guidance increases the effect of the conditioning information and improves sample quality. With guidance = 0, our model produces

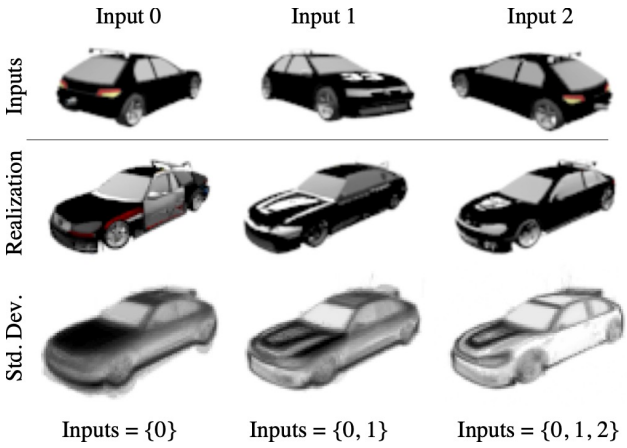

Figure A3: Effect of varying the number of input views. Increasing the number of input views reduces uncertainty, decreasing the pixel-wise standard deviation in novel renderings. Dark pixels in the third row represent higher standard deviation and indicate greater variation in the realizations.

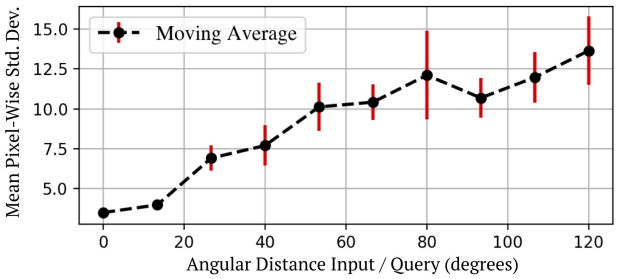

Figure A4: Average pixel variance of generated views vs. the distance between the query camera and the input camera. Input views close to the camera are valuable—the model can directly observe many of the details it must transfer to the output rendering. Input views distant to the camera are more ambiguous—the model is tasked with generating large parts of the rendering from scratch. As the conditioning information gets increasingly ambiguous, novel views get increasingly diverse. Pixel variance is calculated across 50 renderings per pose. Red bars indicate the empirical standard deviation of the moving average.

greater variation of generated views (note the different realizations of the passenger-side door). However, we would consider some of these realizations to be unlikely given the input. Increased CFG strength narrows the distribution of possible outputs, and while we would consider such a set of realizations to be less diverse, each one is of high fidelity. Excessively high guidance strength begins to introduce artifacts and color satura-

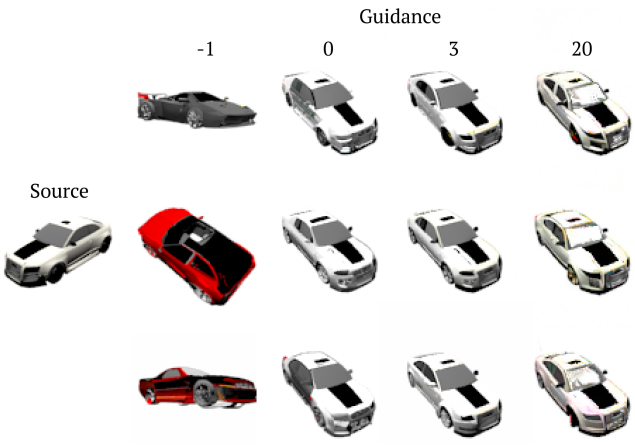

Figure A5: Independent (single-frame input) NVS with various classifier-free guidance (CFG) strengths. For each level of CFG, we show three realizations. With guidance = 0, we sample a "diverse" set of novel views, each plausible, but with variations (e.g. doors). Higher guidance strength reduces diversity but improves sample quality. Excessively high guidance begins to introduce saturation and visual artifacts. Negative guidance upweights the unconditional contribution; with guidance = −1, generation is unconditional.

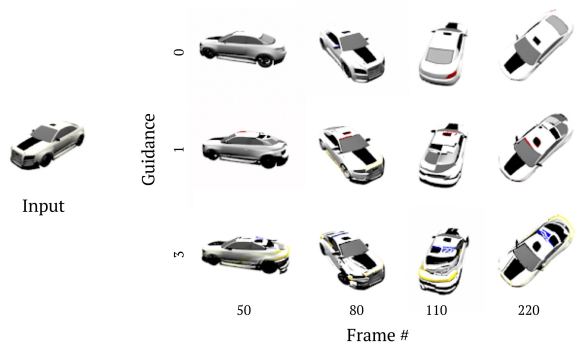

Figure A6: Autoregressive sequence generation with varying CFG strength. With low guidance, we can generate extended autoregressive sequences with little deterioration over time. Higher guidance tends to carry over errors from previous frames, which gradually degrades the quality of subsequent generations.

tion. Negative guidance upweights unconditional prediction; guidance = −1 produces unconditional samples without influence from the input image. In general, when making independent novel view predictions, we find moderate levels of CFG to be beneficial. However, as described in Sec. A1.6, CFG has an adverse effect on the quality of autoregressively generated sequences. As a default, we refrain from using CFG in our experiments.

A1.6. Extended autoregressive generation

Fig. A6 shows autoregressively generated sequences made with varying levels of classifier-free guidance. When making long autoregressive sequences, the ability to suppress errors and return to the image manifold is an important attribute. Unchecked, gradual accumulation of errors could lead to progressive deterioration in image quality. Intuitively, unconditional samples do not suffer from error buildup, since unconditional (CFG = −1) samples make use of *no* information from previous frames. On the other end of the spectrum, highly conditioned (CFG >> 0) samples should be more likely to suffer from error accumulation because they *emphasize* information from previous frames. A happy medium between these two extremes allows the model to use information from previous frames while preventing undesired error accumulation. Empirically, we find that while small positive guidance can reduce frame-to-frame flicker, it enhances the model's tendency to carry over visual errors from previous frames. We observe saturation buildup and artifact accumulation to be significant roadblocks to using CFG when synthesizing long video sequences. For these reasons, we default to using CFG = 0, which we found to enable autoregressive generation of long sequences without significant error accumulation. A solution that enables higher CFG weights for autoregressive generation may make a valuable contribution in the future.

A1.7. Alternative autoregressive conditioning schemes

Baseline strategy When generating a sequence autoregressively, there are many possible strategies, each with a set of tradeoffs. To produce the visual results presented in our work, we used the following baseline strategy, with minor variations for different datasets. As described in the main paper, our baseline strategy is to condition our model on the input image(s), the most recently generated rendering, and five previously generated images, selected at random.

For Matterport3D, when generating long sequences, we select the five previously generated frames from a set of only the 20 most recently generated frames; we additionally condition on every 15^{th} previously generated frame.

For CO3D, we use the two-pass conditioning method discussed below to improve temporal consistency.

Alternative strategies and tradeoffs As described in the main paper, our baseline autoregressive

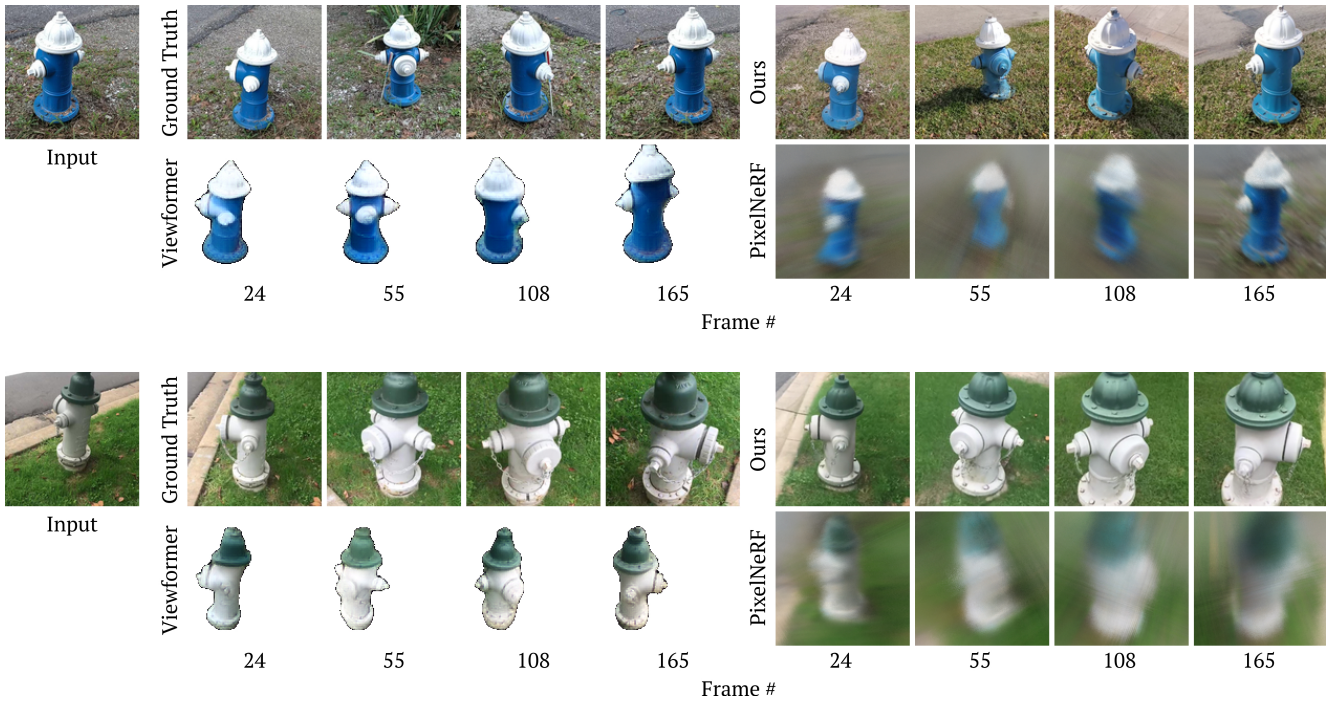

Figure A7: Qualitative comparison for single-view novel view synthesis on CO3D [19] *Hydrants*.

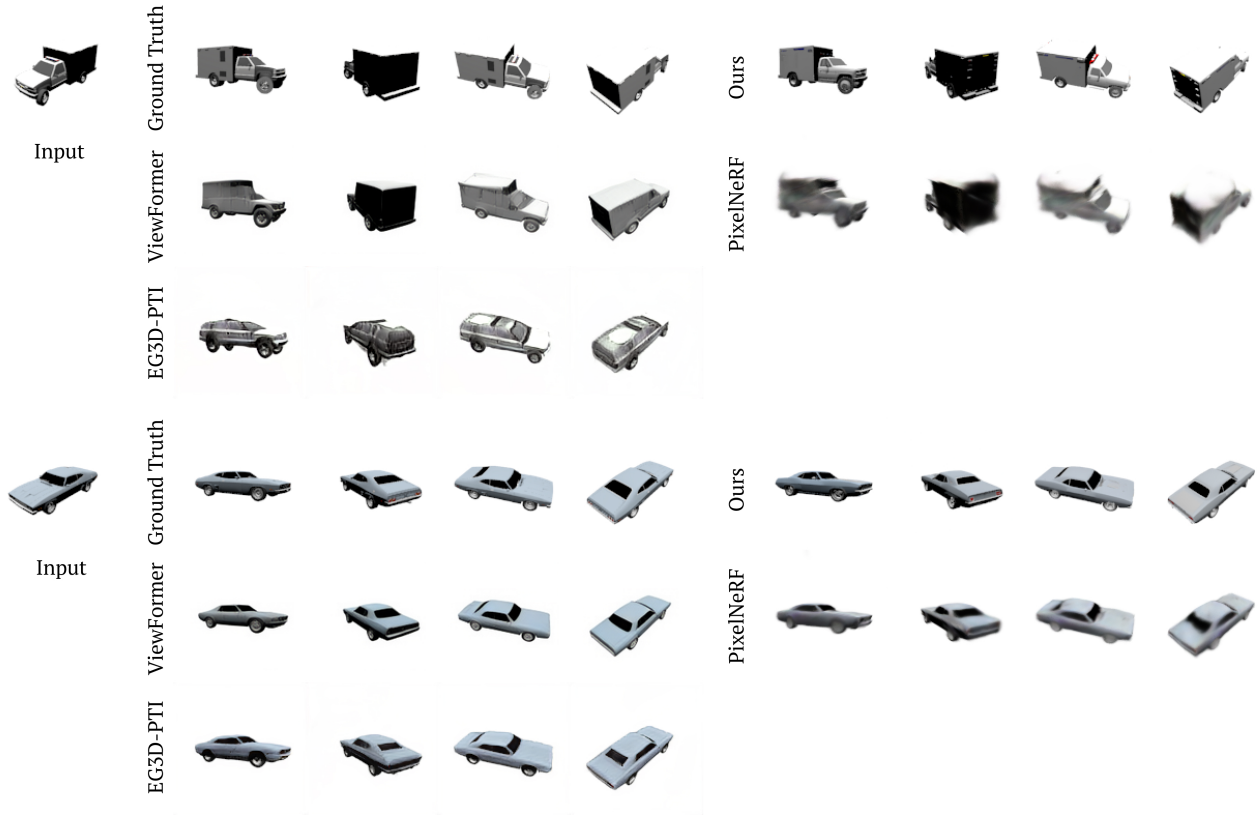

Figure A8: Additional qualitative comparisons against baselines on ShapeNet [2].

Figure A9: Additional samples from our method for short-range NVS on the Matterport3D Dataset [1].

strategy can induce noticeable flickering. One way to reduce flickering is to condition on *only* the previous frame. Doing so almost completely eliminates frame-to-frame flicker. However, this strategy sacrifices long-term consistency and does little to prevent drift; new renderings might not be consistent with frames rendered at the start of the sequence. By contrast, to promote long-term consistency, one could avoid conditioning on previously-generated frames at all and instead condition on only the input image(s). Because drift is the result of error accumulation from conditioning on previous generations, this strategy eliminates potential for drift. However, it suffers from short-term inconsistency (i.e. frame-to-frame flicker). We found our baseline strategy, which conditions on the inputs, the most recent rendering, and several previous renderings, to be a good compromise between long-term and short-term consistency. The number of previously generated images we condition upon affects the behavior. Because we equally weight the contribution of all images we condition upon, increasing the number of previous renderings (which are sampled uniformly from the generated sequence) reduces the relative contribution of the most recent rendering. Increasing the size of this "buffer" of previously-generated conditioning images thus improves long-term consistency at the cost of short-term consistency; reducing the size of the buffer has the opposite effect.

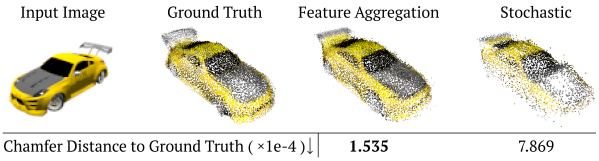

Figure A10: Our default autoregressive conditioning strategy, which aggregates information from multiple views within a feature volume, typically performs at least on par with stochastic view conditioning [28] in geometric consistency, but requires many fewer steps of diffusion to remain effective. Here, we compare COLMAP reconstructions of a sequenced produced by feature aggregation, using 25 steps of denoising, against a sequence produced by stochastic conditioning, using 256 steps of denoising.

One way to suppress flickering is to generate frames in two passes, where in the second pass, we condition on the nearby frames from the first pass in a sliding window fashion. Empirically, conditioning on only the nearest 4 frames during the second pass results in videos with reduced flicker, at the expensive of higher inference computation. However, unless otherwise noted, we render all videos shown with our baseline autoregressive strategy, i.e. *without* these alternative methods.

A1.8. Stochastic Conditioning

To demonstrate the effectiveness of our autoregressive synthesis method, which aggregates conditioning feature volumes from autoregressively selected generated images, we compare to an adaptation of the stochastic conditioning method proposed in 3DiM [28]. We adapt the stochastic conditioning method to our architecture by replacing the feature volume aggregation from autoregressively selected generated images with a single feature volume generated from an image randomly sampled from all previously generated images. As done in 3DiM, the number of diffusion denoising steps is increased significantly and the randomly sampled image is varied at each individual step of denoising. Each generated final image is then added to the set of all previous images and can be used as conditioning

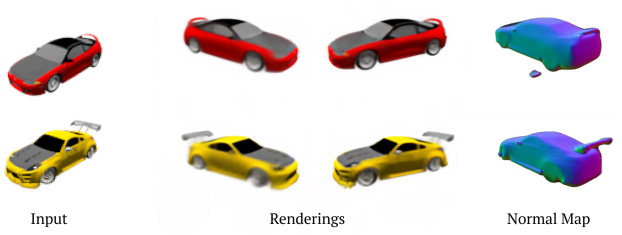

Figure A11: If perfect view consistency is necessary, one can combine our trained diffusion model with Score Distillation Sampling [18] to obtain view-consistent radiance fields.

in subsequent view generations. This alternative form of conditioning is also able to provide the model with information from many generated views, but they are processed independently with each step of denoising, rather than together after a feature volume aggregation.

In Fig. A10, we show 3D reconstruction results from sequences of images generated by our autoregressive synthesis method and with our adaptation of stochastic conditioning [28]. Here, we find that our autoregressive synthesis method performs slightly better than stochastic conditioning in terms of 3D consistency of generated frames as seen by the COLMAP 3D reconstruction and corresponding Chamfer distance. Additionally, we are able to generate novel views significantly faster – in practice, stochastic conditioning requires 256 denoising steps to generate each novel view while our method only requires 25, leading to a $10x$ improvement in speed.

A1.9. 3D Neural Field Fitting with Score Distillation Sampling

Fig. A11 demonstrates that we can fit view-consistent neural radiance fields [15] using our trained diffusion model through Score Distillation Sampling [18]. The application of Score Distillation Sampling to an image-conditioned diffusion model has been explored in recent work [14, 6, 33]. We base our implementation of this experiment on top of Threestudio [7].

A1.10. Additional Common Objects in 3D results

We provide additional results for single-view novel view synthesis (NVS) with real-world objects for CO3D *Hydrants* in Fig. A7. We compare against ViewFormer [13], which has demonstrated success in few-shot NVS on CO3D, and PixelNeRF [30]. However, we note that ViewFormer is not a 1:1 comparison for two reasons: 1. ViewFormer operates with object masks, whereas our method operates with backgrounds. 2. ViewFormer train/test splits did not align with other methods. For this figure, and for comparison videos, we selected objects that were contained in our *test* split but were part of ViewFormer's *train* split. Despite these disadvantages, our method demonstrates a compelling ability to plausibly complete complex scenes.

A1.11. Additional ShapeNet results

Fig. A8 provides additional visual comparisons on the ShapeNet [2] dataset against baselines. In general, our method renders images with sharper details and higher perceived quality than PixelNeRF, while better transferring details from the input image than ViewFormer and EG3D. In this figure, renderings from our method are selected from autoregressively-generated sequences.

A1.12. Additional Matterport3D Results

Fig. A9 presents additional trajectories for single-view, autoregressive generation from our model on the Matterport3D [1] dataset.

A2. Implementation details

We implemented our 3D-aware diffusion models using the official source code of EDM [11], which is available at https://github.com/NVlabs/edm. Most of our training setup and hyperparameters follow [11]; the exceptions are detailed here.

Feature volume encoder, T. Our encoder backbone is based on DeepLabV3+ [3]. We use a Pytorch reimplementation [10] available at https://github.com/qubvel/segmentation_models.pytorch, and ResNet34 [8] as the encoder backbone. We found unmodified DeepLabV3+, to struggle because the output branch contains several unlearned, bilinear upsampling layers; this resolution bottleneck makes it difficult to effectively reconstruct fine details from the input. We replace these unlearned upsampling layers with learnable convolutional layers and skip connections from previous layers. We disable batchnorm and dropout throughout the feature volume encoder. The feature volume encoder expects as input a $3 \times 128 \times 128$ image; it produces a $(16 \times 64) \times 128 \times 128$ feature image, which we reshape into a $16 \times 64 \times 128 \times 128$ volume.

Multiview aggregation. We aggregate information from multiple input views by predicting a feature volume $\boldsymbol{W}_i$ for each input image independently, projecting the query point into each feature volume, sampling a separate feature vector from each feature volume,

and mean-pooling across the sampled feature vectors to produce a single aggregated feature. We experimented with two alternative aggregation strategies: 1. max-pooling, and 2. weighted average pooling, where the feature volumes have an additional channel that is interpreted as a weight by a softmax function. We found these alternative aggregation strategies to perform similarly to mean-pooling.

MLP, f. We use a two-layer ReLU MLP to aggregate features drawn from multiple input images. Our MLP has an input dimension of 16, two hidden layers of dimension 64, and an output dimension of 17, which is interpreted as a 1-channel density τ and a 16-channel feature $\boldsymbol{c}$. We additionally skip the MLP's input feature to the output feature.

Rendering. We render feature images from the model using neural volume rendering [15] of features [16], from the neural field parameterized by the set of feature volumes $\boldsymbol{W}$ and the MLP f. For computational efficiency, we render at half spatial resolution, i.e. 64×64 and use bilinear upsampling to produce a 128×128 feature image. We use 64 depth samples by default, scattered along each ray with stratified sampling. We do not use importance sampling.

UNet, U. The design of U is based on *DDPM++* [27], using the implementation and preconditioning scheme of [11]. U accepts as input 19 total channels (a noisy RGB image, plus a 16-channels feature rendering) of spatial dimension 128^2. It produces a 3-channel 128^2 denoised rendering. For experiments shown in the manuscript, our models contain five downsampling blocks with channel multipliers of [128, 128, 256, 256, 256]. As in [27], we utilize a residual skip connection from the input to U to each block in the encoder of U.

Training. We use a batch size of 96 for all training runs, split across 8 A100 GPUs, with a learning rate of 2×10^{-5}. During training, we sample the noise level σ according to the method proposed by [11] by drawing σ from the following distribution:

$$\log(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2). \tag{1}$$

We use $P_{\text{mean}} = -1.0$, $P_{\text{std}} = 1.4$. During training, we randomly drop out the conditioning information with a probability 0.1 to enable classifier-free guidance. In place of the rendered feature image, we insert random noise.

Our dataset is composed of posed multi-view images, where for each training image, we are given the 4×4 camera pose matrix, the camera field of view, and a near/far plane. For all experiments, we specify a global near/far value for each dataset, where the values are chosen such that a camera frustum with the chosen near/far planes adequately covers the visible portion of the scene. For ShapeNet, near/far $= (0.8, 1.8)$; for MP3D, near/far $= (0., 12.5)$; for CO3D, near/far $= (0.5, 40)$. We found our method to be fairly robust to the chosen values of near/far planes.

For ShapeNet, we train until the model has processed 140M images, which takes approximately 9 days on eight A100 GPUs. For MP3D, we train for 110M images, which takes approximately 7 days on eight A100 GPUs. For CO3D, we train for 170M images, which takes approximately eleven days on eight A100 GPUs.

Augmentation. During training, we introduce two forms of augmentation. First, with probability 0.5, we add Gaussian white noise to the input images. For input images in the range $[-1, 1]$, we sample the standard deviation of the added noise uniformly from $[0, 0.5]$. Second, we apply non-leaking augmentation [11] to U. With probability 0.1, we apply random flips, random integer translations (up to 16 pixels), and random 90^{o} rotations, where the transformations are applied to the input noisy image, the input feature image, and the target denoised image. We condition U with a vector that informs it of the currently applied augmentations; we zero this vector at inference.

Inference. We use the deterministic second order sampler proposed in [11] at inference. As a default, we use $N = 25$ timesteps, with a noise schedule governed by $\sigma_{\text{max}} = 80$, $\sigma_{\text{min}} = 0.002$, and $\rho = 7$, where ρ is a constant that controls the spacing of noise noise levels. The noise level at a timestep i is given in Eq. 2:

$$\sigma_{i<N} = \left({\sigma_{\text{max}}}^{\frac{1}{\rho}} + \frac{i}{N-1} \left({\sigma_{\text{min}}}^{\frac{1}{\rho}} - {\sigma_{\text{max}}}^{\frac{1}{\rho}} \right) \right). \tag{2}$$

Rendering an image from scratch with 25 denoising steps takes approximately 1.8 seconds per image at inference on an RTX 3090 GPU.

"Production" settings for CO3D. For rendering videos of CO3D, we use more computationally expensive "production" hyperparameters to obtain better image quality. Seeking better image quality and detail, we use 256 denoising steps instead of the default 25 denoising steps. Seeking better temporal consistency, we increase the number of samples per ray cast through

the latent feature field, from 64 to 128; we also use the two-pass form of autoregressive conditioning described in Sec. A1.7.

A3. Experiment details

A3.1. Evaluation details

FID Calculation. We compute FID by sampling 30,000 images randomly from both the ground truth testing dataset and corresponding generated frames. We use an inception network provided in the StyleGAN3 [12] repository for computing image features.

KID Calculation. We compute KID by sampling all images from both the ground truth testing dataset and corresponding generated frames. We use the implementation of *clean-fid* [17], available at https://github.com/GaParmar/clean-fid.

COLMAP Reconstructions. We compute COLMAP reconstructions using frames from rendered video sequences. We provide the ground-truth camera pose trajectory as input for all reconstructions. For ShapeNet evaluations, we additionally compute masks by thresholding images to remove white pixels. We leave all settings at their recommended default.

Chamfer Distance Calculation. For all datasets, we compute the bi-directional Chamfer distance between the reconstructed point cloud from synthesized images to the reconstructed point cloud from ground truth images. Additionally, for CO3D, we translate and scale the reconstructed point clouds to lie within the unit cube.

A3.2. Baselines

PixelNeRF [30]. We compare to PixelNeRF for the ShapeNet and CO3D single-image novel view synthesis benchmark. For ShapeNet, we use the official implementation and pre-trained weights for single-category (car), single-image, ShapeNet novel view synthesis evaluation provided at: https://github.com/sxyu/pixel-nerf. We follow the protocol described in the original PixelNeRF paper and SRNs [26] for data pre-processing. We use the provided dataset and splits in the PixelNeRF repository for training and testing of both our method and PixelNeRF (this dataset is slightly different from that used in the SRNs paper due to a bug; see PixelNeRF supplementary information). We follow the same protocol for evaluation as we do for our method and SRNs: view 64 is used as input, and the remaining 249 views are synthesized conditioned on this. For CO3D, we train PixelNeRF from scratch using our train/test splits and using the recommended hyperparameters.

ViewFormer [13]. We compare to ViewFormer on the ShapeNet single-image novel view synthesis benchmark and qualitatively on single-image novel view synthesis for CO3D. We received the data and results for single-image novel view synthesis for the entire ShapeNet testing set from the authors. We compute metrics using their provided ground truth data and synthesized results. The training and testing splits are the same as those used in our method and in PixelNeRF. They use the previously introduced protocol for single-image novel view synthesis evaluation: view 64 is used as input, and the remaining 249 views are synthesized conditioned on this. For CO3D, we instead condition on the first frame from each shown sequence, and generate a video based on this conditioning information. We use provided source code from the official repository at: https://github.com/jkulhanek/viewformer. We do not generate quantitative metrics, as ViewFormer operates on masked and center-cropped images. Additionally, the images, which we use for comparison are in the training set for ViewFormer, while for our method they are in the test set.

Look Outside the Room [20]. We compare against Look-outside-the-room (LOTR), the current state-of-the-art method on novel view synthesis on Matterport3D (MP3D)[1] and RealEstate10K [32] datasets. For LOTR, we obtained the pretrained weights for the MP3D dataset from their official codebase https://github.com/xrenaa/Look-Outside-Room. We match LOTR's data preparation methodology, including identical train/test splits, and we use LOTR's implementation for generating multi-view images from MP3D RGB-D scans. For testing their method, we prepare a common set of 200 input images from the test split with the trajectories and ground truth images for the next 10 frames for each input. Then, we run the LOTR method on the given input using the code from their Github repository, using 3 overlapping frame windows, as stated in their paper. We run LOTR on the next 10 frames, given the input frame, and measure the metrics against the ground truth.

Additional Baselines for MP3D To further evaluate our method's effectiveness on the novel-view synthesis task on MP3D scenes, we compare against additional baselines of GeoGPT [22] and SynSin [29]. Note that these two baselines, along with another recent work of PixelSynth [21], have been already shown to

underperform against LOTR [20]. Since GeoGPT does not provide pre-trained models or rendered images for MP3D, we inquired the authors of LOTR for the images they used for the benchmarks. The acquired NVS images of GeoGPT and SynSin are rendered by the exact same protocol as our experiments, except that they proceeded five frames from the initial input images for 200 sequences (thus we have 1,000 images in total). We note that the trajectories used for these acquired images are different from the trajectories we used for our experiments because the trajectories are generated randomly via the Habitat embodied agent simulation [24]. However, at 1000 trajectory samples, we believe our comparisons are statistically significant. The final numbers we computed show similar trends to those reported in the LOTR paper, further confirming the validity of the comparisons. Both qualitatively and quantitatively, we observe that our novel-view renderings are significantly more desirable.

A3.3. Dataset details

ShapeNet [2]. We extensively evaluate our method on the ShapeNet dataset. The full ShapeNet dataset contains different object categories, each with a synthetically generated posed images in pre-defined training, validation, and testing sets. In our work, we specifically evaluate with the "cars" category, and focus on single-image novel view synthesis. We use the version of the dataset provided in PixelNeRF [30] for consistency in training and evaluation, keeping all frames in the dataset at 128^2 resolution and doing no additional pre-processing. As described in the main paper, the training set contains 2,458 cars, each with 50 renderings randomly distributed on the surface of a sphere. The test split contains 704 cars, each with 250 rendered images and poses on an Archimedean spiral. During the training of our method, we use the defined training split, randomly sampling between one and three input frames with the objective of synthesizing a randomly selected target frame for a specific object instance. In evaluation, we use the defined testing split, use image number 64 as input, and synthesize the other 249 ground truth images. We note that since these images are synthetically generated at only 128^2, they lack backgrounds and fine detail. However, the accuracy of poses in the constrained environment and consistent evaluation method between baselines allows for easily providing quantitative benchmarks for single-image novel view synthesis.

Matterport3D [1]. We showcase our algorithm on a highly complex, large-scale indoor dataset, Matterport3D (MP3D). MP3D contains RGB-D scans of real-world building interiors. Scenes are calibrated to metric scale, and thus there is no scale ambiguity. We preprocess MP3D scans into a dataset of posed multi-view images following the procedure detailed in LOTR [20] and SynSin [29]. Specifically, we generate the image sequences by simulating a navigation agent in the room scans, using the popular Habitat [24] API. We randomly select the start and end position within the MP3D scenes and simulate the navigation towards the goal via Habitat. The agent is only allowed to take limited actions, including going forward and rotating 15 degrees. During training, we randomly sample a target frame and then select 1 to 3 random source frames in the neighborhood of 20 frames for conditioning.

Common Objects in 3D [19]. We validate our method on a real-world dataset: Common Objects in 3D (CO3D). The CO3D dataset consists of several categories. We train on CO3D Hydrants, which contains 726 scenes. The average scene consists of around 200 frames of RGB video, object masks, poses, and semi-sparse depth. We note that the CO3D dataset is quite unconstrained: even across scenes within a category, aspect ratio, resolution, FOV, camera trajectory, object scale, and global orientation all vary. Additionally, we note that the dataset is noisy, with several examples of miscategorized objects and numerous extremely short or low-quality videos. Such noise adds to the challenge of single-image NVS.

In preparing data, we first center-crop to the largest possible square, then resize to 128^2 using Lanczos resampling. We adjust the camera intrinsics to reflect this change. We also seek to normalize the canonical scale of scenes across the dataset. To do so, we examine the provided depths within each scene, and consider the depth values that fall within the object segmentation mask. For each image, we calculate the median value of the masked depth. Taking the mean of these median values across the scene gives us a rough approximation of the distance between the camera and object. We adjust the scale of the scene so that this camera-object distance is identical across every scene in the dataset.

To help resolve scale, which is highly variable across the dataset, and to provide information parity with PixelNeRF, which has access to a global reference frame, we provide our feature encoder, T, with the location of the global origin. In addition to each input RGB image, we concatenate a channel that contains a depth rendering of the three coordinate planes, as rendered from the input camera. We modify T to accept the four-channel input. We find this input augmentation to improve our model's ability to localize objects.

A4. Discussion

A4.1. Alternative approaches

GAN-based generative novel view synthesis. We have presented a diffusion-based generative model for novel view synthesis, but in principle, it is possible to construct a similar framework around other types of generative models. Generative Adversarial Networks [5] (GANs), are a natural fit, and adversarial training could drop in to replace our diffusion objective with minor changes. While recent work [4] has demonstrated that diffusion models often outperform GANs in mode coverage and image quality, GANs have a major advantage in speed. Future work that aims for real-time synthesis may prefer a GAN-based 3D-aware NVS approach.

Transformer-based, geometry-free multi-view aggregation strategies. A promising alternative to explicit geometry priors, such as the type we have presented in this work, is to instead make use of powerful attention mechanisms for effectively combining multiple observations. Scene Representation Transformers [23] utilize a transformer-based approach to merge information from multiple views, which is effective for NVS on both simple and complex scenes. We explored an SRT-based variant of Γ, which would forego explicit geometry priors for a transformer and light field [25] based conditioning scheme. However, we had difficulty achieving sufficient convergence and in justifying the additional compute cost. Nevertheless, related approaches could be a promising area for future study.

A4.2. Limitations

We believe our method to be a valuable step towards in-the-wild single-view novel view synthesis but we acknowledge several limitations. While we demonstrate our method to be competitively geometrically consistent, it is not inherently 3D or temporally consistent. Noticeable flicker and other artifacts are sometimes visible in rendered sequences.

While our model generally produces plausible renderings, it may not always perfectly transfer details from the input. On ShapeNet, this sometimes manifests as an inability to replicate the angle of car tires or the style of windows across the line of symmetry; on more complex datasets, the model sometimes struggles to transfer fine details. We use a relatively lightweight, ResNet-backed Deeplab feature encoder. A more powerful encoder, potentially one that makes use of attention to improve long-range information flow, may resolve these issues.

References

[1] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 5, 6, 8, 9

[2] 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. 4, 6, 9

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 6

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 10

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 10

[6] Jiatao Gu, Alex Trevithick, Kai-En Lin, Josh Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion. In *International Conference on Machine Learning*, 2023. 6

[7] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. https://github.com/threestudio-project/threestudio, 2023. 6

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[9] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 2

[10] Pavel Iakubovskii. Segmentation models pytorch. https://github.com/qubvel/segmentation_models.pytorch, 2019. 6

[11] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 6, 7

[12] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Proc. NeurIPS*, 2021. 8

[13] Jonáš Kulhánek, Erik Derner, Torsten Sattler, and Robert Babuška. ViewFormer: NeRF-free neural rendering from few images using transformers. In *European Conference on Computer Vision (ECCV)*, 2022. 6, 8

[14] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 6

[15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 6, 7

[16] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11453–11464, 2021. 7

[17] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022. 8

[18] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 6

[19] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3D: Large-scale learning and evaluation of real-life 3D category reconstruction. In *IEEE International Conference on Computer Vision (ICCV)*, pages 10901–10911, 2021. 4, 9

[20] Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing a consistent long-term 3D scene video from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3563–3573, 2022. 8, 9

[21] Chris Rockwell, David F Fouhey, and Justin Johnson. PixelSynth: Generating a 3D-consistent experience from a single image. In *IEEE International Conference on Computer Vision (ICCV)*, pages 14104–14113, 2021. 8

[22] R. Rombach, P. Esser, and B. Ommer. Geometry-free view synthesis: Transformers and no 3D priors. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 8

[23] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 10

[24] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347, 2019. 9

[25] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 10

[26] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 8

[27] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 7

[28] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. *arXiv preprint arXiv:2210.04628*, 2022. 5, 6

[29] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7467–7477, 2020. 8, 9

[30] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2021. 6, 8, 9

[31] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[32] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ACM Transactions on Graphics (SIGGRAPH)*, 2018. 8

[33] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. In *CVPR*, 2023. 6