Novel View Synthesis with Pixel-Space Diffusion Models

Supplementary Material

A. Implementation Details

Our code implementation for both evaluation and training, along with model weights, are available at https://github.com/apple/ml-vivid.

A.1. Model Architecture & Training

Our architecture is based on the U-Net [31] outlined in EDM2 [17]. To facilitate the processing of both the source image and the noisy target, we duplicate our U-Net architecture into two dedicated networks, an encoder and a decoder, as outlined in subsection 3.1 and Figure 2. Specifically, we employ joint-attention between the encoder's deep features and the decoder's, in which keys and values are computed for both the encoder and decoder stream and concatenated, whereas queries are computed only from the decoder. We find empirically that joint-attention performs better than self-attention followed by cross-attention. We use the default 3 layer per resolution, using resolutions $[64, 32, 16, 8]$ and $[256, 128, 64, 32]$ for the base and SR models respectively. Beyond the existing attention layers, we add a single attention layer at the second-highest resolution (32) on the base model to enhance fine detail transfer between encoder and decoder through the attention mechanism.

We condition both the encoder and decoder networks on the diffusion timestep. If time conditioning is omitted from the encoder network, it would only need to be run once for each generation, as the source image and geometry do not change. In preliminary experiments we find that this change has a slight negative effect on the results, as expected due to the encoder's lack of ability to match its extracted features to the current diffusion timestep. For simplicity, we focus on the variant conditioning both networks on the timestep, seeking higher quality results, and leave the exploration of the more efficient option to future work.

We base our hyperparameter choices on EDM2 [17], making no change across our experiments to: number of residual blocks, channel multiples, sampling algorithm, noise schedule, number of inference diffusion steps, and Adam optimizer hyperparameters. We do not use dropout and use training noise level distribution with a log-normal distribution where mean and standard deviation are $P_{mean} = -0.8$ and $P_{std} = 1.6$ respectively. Table 4 lists our experiment-specific training hyperparameter choices.

During training, we iterate over scenes in the RealEstate10K dataset, uniformly choosing two views as the source and target views regardless of the distance between the frames. During inference, we first uniformly sample the source view and later uniformly sample a target view

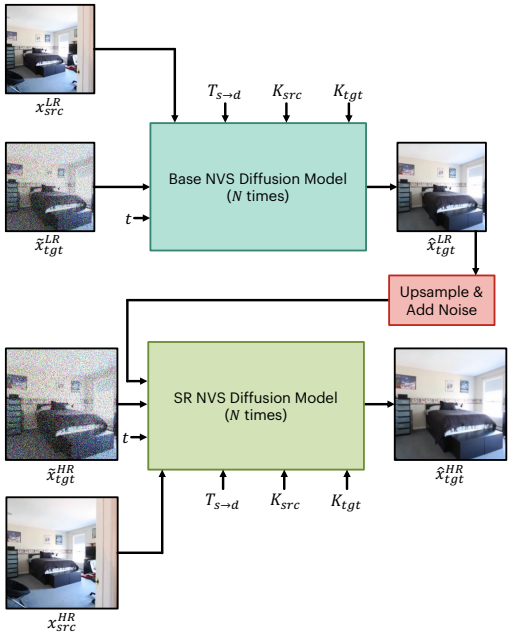

Figure 9. End-to-end inference using our base and super-resolution (SR) NVS diffusion models.

that is 30-60 (mid) or 60-120 (long) frames apart. Source views that do not have possible targets at the appropriate frame distance are filtered out. We have noticed that tailoring the distance between frames to 30-120 (the relevant distance) during training is not beneficial for image quality or metrics.

For sampling with the base model, we employ Classifier-Free Guidance [11] (CFG) using a separate unconditional diffusion model, following EDM2 [17], instead of the more commonly used single model with label dropout. The unconditional model does not contain the encoder network, instead receiving zeros as pose encoding and joint-attention features. We choose a CFG coefficient of 1.5 for generating samples from RealEstate10K, and 2.0 for the experiments in section 4. The EMA values are not tuned further for CFG.

A.2. Cascaded Diffusion and Super-Resolution

As detailed in subsection 3.1, we use a cascaded diffusion model design to retain the pixel-space information used in NVS. Thus, a super-resolution (SR) model is trained separately from the base model. Specifically, the task of NVS super-resolution is a relatively simple task, having both the low frequency information from the low-resolution conditioning image and the high-frequency semantics and texture from the source view. For this reason, we opt to use an SR

Model	Batch Size	Training Iterations	Base Channel Width	Learning Rate	EMA
Base (Ablation)	512	2^{18}	128	0.012	-
Base (Final)	1024	2^{20}	128	0.012	0.1
Base (For CFG)	1024	2^{19}	128	0.012	0.05
SR	256	2^{20}	64	0.01	0.05

Table 4. Training Hyperparameters.

model with smaller capacity and batch size, and a shorter training time. Additionally, to alleviate the distribution gap between ground-truth and generated low-resolution conditions, we add white Gaussian noise to the conditioning image before using it as an input to the SR model, as proposed in [13]. We find that noise of standard deviation 0.25 works well. Our end-to-end inference system (including the base and SR models) is depicted in Figure 9.

While our SR model in this work was limited to 256×256 images due to the constrains of available data, our method could be extended to even higher resolutions. This could be doen by training an alternative SR model from 64×64 to a new high resolution, or training additional SR models as needed. As described above, the SR models are relatively small compared to the base model, and should scale in a straightforward manner to higher resolutions.

B. Geometry Encoding Ablation Details

B.1. Pose Embedding

To apply our pose embedding, we encode the camera information into a 20-element encoding, as such:

$$E_{pose} = \texttt{concat}\left(\texttt{flatten}\left(\boldsymbol{T}_{t\to s}\right), f_{src}, p_{src}, f_{tgt}, p_{tgt}\right) \tag{1}$$

Where f and p are the focal length and principle point extracted from the intrinsic matrixes $\boldsymbol{K}_{src}$ and $\boldsymbol{K}_{tgt}$. The embedding conditions each U-Net block using a single fully-connected layer and summation, similarly to class embedding conditioning in EDM2 [17]. The embedding is normalized to zero mean and standard deviation of one using precomputed estimations based on 64×64 source-target pairs from RealEstate10K. For the 256×256 SR model the embedding is altered by extrapolating the statistics of 64×64 intrinsics to 256×256 intrinsic matrices. The extrinsics do not change for the different image resolutions.

B.2. Epipolar Attention Bias

We test the use of epipolar attention bias as a type of geometric encoding. Our implementation is inspired by the epipolar attention used in [44], in which an epipolar attention matrix is computed by using a soft cutoff function (sigmoid) on the epipolar distance between any two pixels in the source and target images. In [44], the epipolar attention matrix multiplies the cross-attention matrices in all layers, effectively zeroing out the feature correlations that do not fit the geometry. We opt to use the epipolar attention matrix as an attention bias instead, replacing the multiplication operator with addition. A soft attention bias enables the network to transfer information between the source and target streams even when the information does not strictly fit the geometric composition, while multiplication by zeroes strictly prohibits that. Furthermore, we allow each attention head to learn its own mixing parameter for the epipolar attention, enabling the network to learn separate heads for strictly geometric correspondence and semantically significant features. Specifically, we learn 4 scalars per head in each attention block, modulating the amplitude (m), temperature (τ), cutoff (c), and bias (b) for mixing the epipolar attention bias. The following formula computes our epipolar attention bias ($\boldsymbol{A}_{epipolar}$) using the epipolar distance matrix ($\boldsymbol{D}_{epipolar}$) and the learnable mixing parameters:

$$\boldsymbol{A}_{epipolar} = m \cdot \sigma(\tau \cdot (c - \boldsymbol{D}_{epipolar})) + b \tag{2}$$

Specifically, the bias is needed due to the use of joint-attention, as the epipolar attention bias is only added to the "cross" part of the attention matrix (*i.e.*, the attention between source and target view features). The epipolar attention mixing parameters are initialized such that all elements of the epipolar attention bias are zero.

B.3. Monocular Depth Estimation

We use the small indoor metric model from DepthAnythingV2 [52] to estimate the source image depth. The depth-map is inverted, normalized by the maximum value and further normalized across the depth map distribution to zero mean and standard deviation of one, to have a similar magnitude to the input source image. While the depth normalization preserves the ordinality of the depth, its metric accuracy is effectively erased. In any case, since the depth prediction is not up to scale with the RealEstate10K poses, the metric accuracy of the MDE is not beneficial for NVS.

B.4. Depth Coordinate Warping

We implement coordinate warping following the method outlined in GenWarp [37]. We start by creating a 2-dimensional grid for the source image's pixels. We then warp (in 3 dimensions) this grid to the target view using the monocular depth predicted with the same model used in the

previous section. As a result, we obtain a target view grid where each pixel points to its matching coordinate from the source view grid. Finally, we encode both the source and target grids into 128 Fourier feature maps, and concatenate them to the encoder and decoder inputs, respectively. This increases the efficiency of the warping operation by skipping the interpolation step, instead relying on the network to learn the correlations.

C. Single-Image Augmentation

C.1. Augmentation Implementation Details

We use the OpenImages v5 dataset [20] for the single-image augmentation experiment. We first center-crop and resize all images to 512×512. Then, we select the augmentation parameters for the source and target views, creating two distinct camera poses and matching warping operations. We uniformly sample the yaw, pitch, and roll[3] as such: (i) with probability 0.5, we sample them from $[-5.5°, 5.5°], [-5.5°, 5.5°], [0°, 0°]$ and center crop to 384×384; and (ii) otherwise, we sample them from $[-8.3°, 8.3°], [-8.3°, 8.3°], [-3.5°, 3.5°]$ and center crop to 320×320. These parameters also determine the simulated geometry, by computing the rotation matrix corresponding to the sampled angles for both the source and target views. The intrinsic matrix is constructed to match the statistics of RealEstate10K images, where the focal length and principle point ate chosen as $(307.2, 307.2)$ and $(256, 256)$ respectively, for the original 512×512 images. Finally, the produced source and target views are resize to the appropriate model resolution, and mixed into the training batches as a constant fraction of every training batch.

C.2. Generalization Quantitative Analysis

We use 20 scenes from the well-established datasets LLFF [22], MipNeRF-360 [1], and Ref-NeRF [47] to perform a qualitative evaluation of the benefit of our proposed single-image augmentation. We use the preexisting camera poses and intrinsics, rescaled for each scene using a single scalar value. As a replacement for the "mid" and "long" distance between frames used in RealEstate10K, we divide the source-target pairs into ranges based on a certain LPIPS [57] threshold, chosen for each scene to mitigate the scene-level scale ambiguity issue.

D. Method Comparison

D.1. Evaluation of Previous Methods

In subsection 3.3, we compare our work to previous NVS methods. While we mostly follow the evaluation strategy outlined in GenWarp [37], their use of data filtering and the lack of explicit data sampling procedures prevent us from performing the exact same evaluation as in previously published results. For this reason, we employ our own source and target view procedures for evaluation (as detailed in Appendix A). We use the officially published code for evaluating GeoGPT [29], PhotoNVS [54], and GenWarp [37]. All models were evaluated on the 256×256 resolution. Because GeoGPT and GenWarp were trained to operate in different resolutions (208×368 and 512×512 respectively) we resize the source center-cropped 360×360 images from RealEstate10K [59] to the relevant resolution before the NVS operation, and then resize again to 256×256 to perform the evaluation. Additionally, GeoGPT uses 3D points (based on COLMAP [34, 35] from the entire scene) to scale its estimated depth input and solve the NVS task. This gives GeoGPT an unfair advantage other alternative methods for which the scale ambiguity problem remains. For GenWarp, we used the publicly available checkpoints, which are not trained exclusively on RealEstate10K, as no RealEstate10K model is publicly available.

D.2. Additional Metrics for Method Comparison

We augment the comparison in Table 2 with 3 additional metrics: (i) Joint FID [5] (JFID), which is a Fréchet distance over concatenated features from both the source and target (ground-truth or generated) images; (ii) Fréchet distance computed in the DINOv2 [25] space following [17, 41], named FDD (Fréchet DINOv2 Distance); and (iii) JFDD (Joint Fréchet DINOv2 Distance), which is the same as JFID but using DINOv2 instead of Inception. The full results are shown in Table 6. The source image is included in the table to provide a frame of reference for the different metrics. We expect the source image to achieve high perceptual quality in the metrics, because the source and target images are both drawn from the same dataset, and thus in theory they should have the same probability distribution.[4] At the same time, the PSNR of the source images is consistently subpar compared to NVS methods, as (obviously) the source views do not solve the NVS task. To quantify the perceptual quality of a generated sample that does solve the NVS task we propose using JFID [5], a metric that attempts to measure similarity of conditional distributions. The source image under-performs all NVS methods in JFID as expected. Our method (named "VIVID (RE10K)" in the table) is superior to all the tested methods in JFID, in line with the improvement in both FID and PSNR, making it state-of-the-art across all metrics in both mid- and long-range. The results shown in FDD and JFDD corroborate our conclusions, and provide more robust perceptual quality metrics [41]. Finally, we also include the version of our model trained on both RealEstate10K and OpenImages as

[3] We use the COLMAP [34, 35] coordinates, where x, y, and z axes point left, down, and forward respectively.

[4] FID is limited here to 10K images, creating a discrepancy between the empirical distributions of source and target views.

GeoGPT [29]	0.23
PhotoNVS [54]	36.12
GenWarp [37]	0.67
VIVID	0.90
VIVID-Fast	0.41

Table 5. Comparison of NVS inference times for different methods, in seconds per image.

"VIVID (RE10K + OI)" in the table, showing that training on single-view data with our proposed 10% augmentation does not degrade performance on RealEstate10K, and even offers a slight improvement.

D.3. Inference and Training Time Comparison

We compare inference training times for the methods discussed in this section. Table 5 shows the sampling time per image, measured by generating 1000 samples on a single A100 GPU. We also test VIVID-Fast, our model with less diffusion steps and no CFG, which maintains similar FID and PSNR to our regular model (superior to all other options).

For training, VIVID training consumed 18k A100 GPU hours. To compare, GenWarp consumed an equivalent of 5k (+151k for Stable Diffusion pretraining). PhotoNVS and GeoGPT did not report compute numbers.

D.4. Comparison of Fine Details

Figure 10 demonstrates the advantage of our model in the retention of fine details, which stems from our use of cascaded source-conditioned diffusion models in the pixel space instead of latent autoencoders. For this experiment, all images were resized to 256×256 before being resized again to the model's input size. This was done to ensure all models receive information in equal resolution, for the fairness of comparing fine details.

Figure 10. Example of fine detail retained by our model, compared to alternative latent-diffusion-based methods.

E. Additional Results

E.1. Generated Videos

In this paper, we do not directly address multi-generation consistency, but rather focus on improving image-to-image NVS performance. That being said, we can still demonstrate effective multi-frame consistency by producing auto-regressive NVS videos (starting from a source frame, and following a multi-frame camera trajectory from the test set). Figure 11 shows several frames of videos generated in this manner, proving that consistency to the source image is maintained.

Figure 11. 2 NVS videos generated with VIVID auto-regressively.

E.2. Additional Generalization Examples

Figure 12 shows additional qualitative examples for generalization with our model on images from OpenImages.

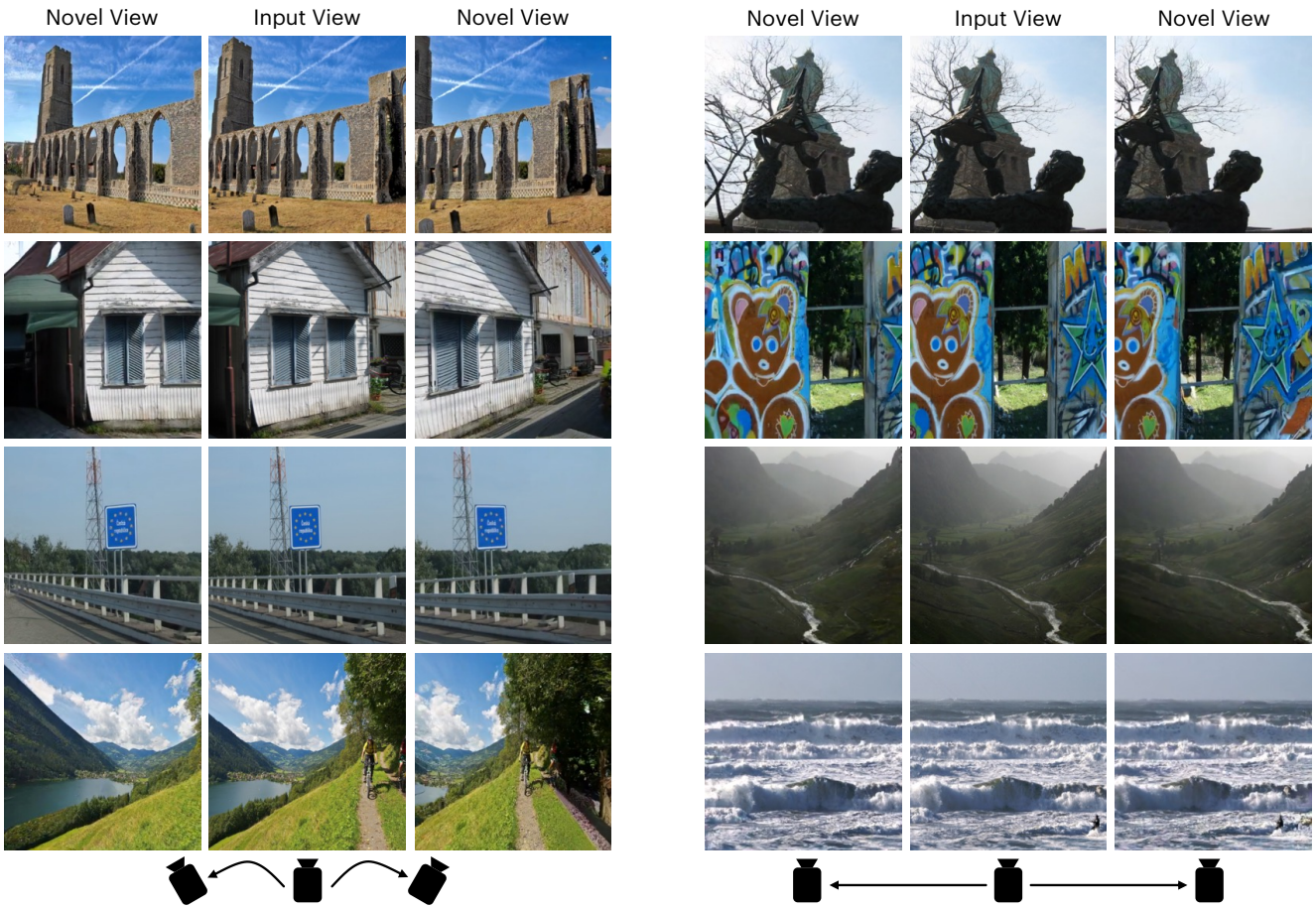

Figure 12. Additional examples for generalization from the OpenImages dataset, exhibiting rotation (left) and translation (right).

Method	Mid-range					Long-range				
	FID ↓	PSNR ↑	JFID ↓	FDD ↓	JFDD ↓	FID ↓	PSNR ↑	JFID ↓	FDD ↓	JFDD ↓
GeoGPT [29]	6.43	14.06	13.19	288.26	444.29	7.22	13.13	13.46	332.49	455.78
PhotoNVS [54]	7.12	13.32	13.62	433.47	559.42	9.22	12.05	15.76	552.42	668.88
GenWarp [37]	5.91	13.43	10.52	68.70	101.39	7.38	12.10	13.62	120.54	171.27
VIVID (RE10K)	<u>2.89</u>	<u>17.36</u>	<u>6.26</u>	<u>41.20</u>	<u>67.43</u>	<u>3.89</u>	**15.21**	<u>8.18</u>	<u>80.44</u>	<u>118.75</u>
VIVID (RE10K + OI)	**2.82**	**17.38**	**6.14**	**38.13**	**63.37**	**3.77**	<u>15.19</u>	**8.02**	**72.62**	**109.53**
Source Image	2.58	13.12	73.76	19.32	528.71	3.00	11.91	51.07	14.09	309.47

Table 6. Comparison to previous methods, including additional metrics. Evaluation is done on 10K source-target pairs from RealEstate10K. Best results in each column are in bold, second best are underlined.