Consistent View Synthesis with Pose-Guided Diffusion Models Supplementary Material

Hung-Yu Tseng[1] Qinbo Li[1] Changil Kim[1] Suhib Alsisan[1] Jia-Bin Huang[1,2] Johannes Kopf[1]
[1]Meta [2]University of Maryland, College Park

1. Overview

In this supplementary material, we first present qualitative results of the existing and our methods. Second, we provide the technical details of the proposed approach to complement the paper. Finally, we describe the details of the experiments, including the dataset pre-processing, evaluation metric computation, and reproducing existing approaches.

2. Qualitative Results

We provide more qualitative results, i.e., *videos* on the project page.[1]

Comparing with previous SOTA methods. We show the single-image view synthesis results of difference scenes generated by the GeoGPT [10], LoR [9], SE3DS [3] and the proposed pose-guided diffusion models.

Generating diverse outputs: We demonstrate that the proposed method is capable of producing multiple realistic videos from the same set of inputs.

Ablation study: We show the qualitative results generated by the Source/target views concatenation and Cross-view attention baselines described in Table 3 in the paper. Specifically, we use different diffusion models to generate the 64×64 videos. We then use the *same* super-resolution diffusion model introduced in Section 3.2 in the paper to get the final 256×256 results. Consistent with the quantitative results shown in Table 3 in the paper, the per-frame quality of the videos generated by the baseline approaches degrades significantly in the final few frames.

Long-range view interpolation: In addition to single-image view synthesis, the proposed method can also *interpolate between two far-away viewpoints*. The key is to leverage the stochastic conditioning approach described in Section 3.3 of the paper. Specifically, to interpolate the view i between two viewpoints l and k, we randomly sample the source view image x^j from the two viewpoints, i.e., $x^j \sim P(\{x^l, x^k\})$ during the backward process in the diffusion model. The sampling probabilities of each input view are determined inverse proportional to the distance to the output viewpoint. In this experiment, we take the 1st, 11th, and 21th frames from each testing video as the input. We interpolate between these input views to obtain a 21-frames video as the result. As no existing approach tackles the long-range view interpolation problem, we compare our approach with an alternative large-motion frame interpolation approach FiLM [8].

Failuer cases. We show several example failure cases. First, the proposed method cannot candle the case where the scene scale is significantly different to those in the training data, e.g., natural scenes. Second, our approach fails to produce high-quality results with rare camera pose sequences, e.g., going up/down stairs.

Flickering caused by super-resolution. To understand the flickering produced by our algorithm, we present the x-t slice visualization in Figure 2. The flickering mainly comes from the per-frame super-resolution stage.

2.1. Technical Details

Epipolar line computation. We present the epipolar line computation in Figure 1. Given a point $\mathbf{p}^i$ on the image plane at the target view i and the relative camera pose $\{\mathbf{K}, \mathbf{R}^{i\to j}, \mathbf{t}^{i\to j}\}$, the goal is to find the corresponding epipolar line on the image plane at the source view j. We first project the point $\mathbf{p}^i$ onto the source view image plane as $\mathbf{p}^{i\to j}$, namely

$$\mathbf{p}^{i\to j} = \pi\big(\mathbf{R}^{i\to j}(\mathbf{K}^{-1}\mathbf{p}^i) + \mathbf{t}^{i\to j}\big), \tag{1}$$

[1] https://poseguided-diffusion.github.io

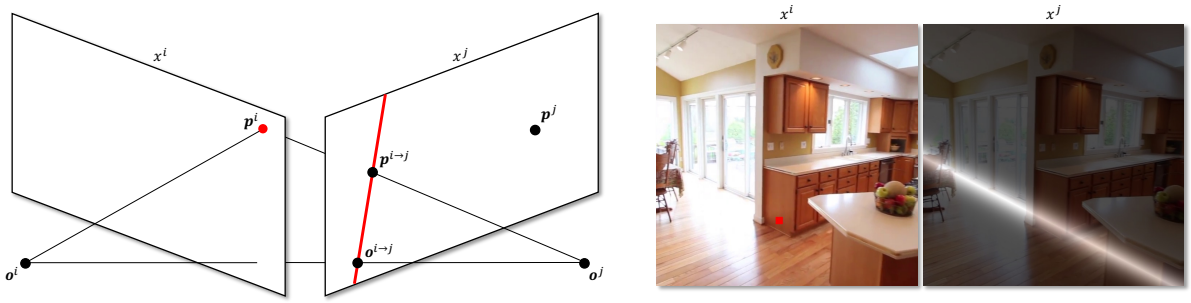

Figure 1. **Epipolar line.** (*left*) We show the computation of the epipolar line (red line) at the source view image x^j which corresponds to the point $\mathbf{p}^i$ (red dot) on the target view image x^i. (*right*) We visualize the epipolar line on the source view image x^j of the point $\mathbf{p}^i$ (red dot) on the target view image x^i. We compute the weight map (right image) according to the epipolar line using (5).

where π is the projection function. We also project the camera origin $\mathbf{o}^i = [0, 0, 0]^T$ at the target view i onto the source view image plane as $\mathbf{o}^{i\to j}$:

$$\mathbf{o}^{i\to j} = \pi\big(\mathbf{R}^{i\to j}(\mathbf{K}^{-1}[0, 0, 0]^T) + \mathbf{t}^{i\to j}\big). \tag{2}$$

Then the epipolar line of the point $\mathbf{p}$ on the source view image plane can be formulated as

$$\mathbf{p}_{\text{epipolar}} = \mathbf{o}^{i\to j} + c(\mathbf{p}^{i\to j} - \mathbf{o}^{i\to j}) \qquad c \in \{-\infty, \infty\} \in \mathbb{R}. \tag{3}$$

Finally, the distance between a point $\mathbf{p}^j$ on the source view image plane and the epipolar line can be computed as

$$d(\mathbf{p}_{\text{epipolar}}, \mathbf{p}^j) = \|(\mathbf{p}^j - \mathbf{o}^{i\to j}) \times (\mathbf{p}^{i\to j} - \mathbf{o}^{i\to j})\| / \|\mathbf{p}^{i\to j} - \mathbf{o}^{i\to j}\|, \tag{4}$$

where $\times$ and $\|\cdot\|$ indicate vector cross-product and vector norm, respectively. According to the epipolar line, we compute the weight map, where higher pixel values indicate closer distance to the line

$$m_{\mathbf{p}^i,\mathbf{K},\mathbf{R}^{i\to j},\mathbf{t}^{i\to j}}(\mathbf{p}^j) = 1 - \text{sigmoid}\big(50(d(\mathbf{p}_{\text{epipolar}}, \mathbf{p}^j) - 0.05)\big) \qquad \forall \mathbf{p}^j \in x^j. \tag{5}$$

We use the constant 50 to make the sigmoid function steep, and use the constant 0.05 to include the pixels that are nearby the epipolar line. An example weight map is visualized in the right-hand side of Figure 1. After estimating the weight maps for all positions in the source view image, we stack these maps and reshape to get the epipolar weight matrix $E_{i,j}$, which is used to compute the epipolar attention described in (6) in the paper. Note that if the epipolar line does not intersect with the target view, we assign the same value for all spatial positions in the epipolar weight matrix $E_{i,j}$. This makes our epipolar attention falls back to the common cross-attention.

Stochastic frame sampling. We randomly sample the source view in both the training and inference stages. We observe inferior results if we use stochastic sampling only during the inference time. We hypothesize this is due to the mismatch of the input relative pose distributions between the training and inference stages.

Super-resolution. We present the super-resolution model details in Figure 3. We first bilinearly up-sample the low-resolution target view, and concatenate it with the noised high-resolution target view. Combining the features extracted from the high-resolution source view, we train the UNet network to de-noise the high-resolution target view. We empirically find that taking the source view as input improves the temporal consistency in high-resolution videos.

Training details. We provide the training details, including the UNet architecture as well as the hyper-parameters in Figure 4 and Table 1. We implement the proposed method with PyTorch [7], and use 8 Nvidia V100 GPUs to conduct the training.

2.2. Experiment Details

Dataset processing. We follow Lai *et al*. [5] to process the RealEstate10K (Re10K) [13] and Matterport3D (MP3D) [1] datasets and split them into training set and test set. Note that the availability of the YouTube videos in the Re10K dataset changes over time. From the available videos, we remove the short videos (less than 200 frames) and eventually obtain $61,986$ videos in the training set and $1,763$ videos in the testing set. We randomly select 500 videos from the testing set for all quantitative and qualitative experiments. As for the MP3D dataset, there are $6,097$ videos in the training set and $1,062$ videos in the test set, and again we select 500 videos from the test set for evaluation.

Figure 2. **X-T slice visualization.** We visualize the x-t slice of the low-resolution (*top*) and high-resolution (*bottom*) videos. The flickering mainly comes from the super-resolution stage.

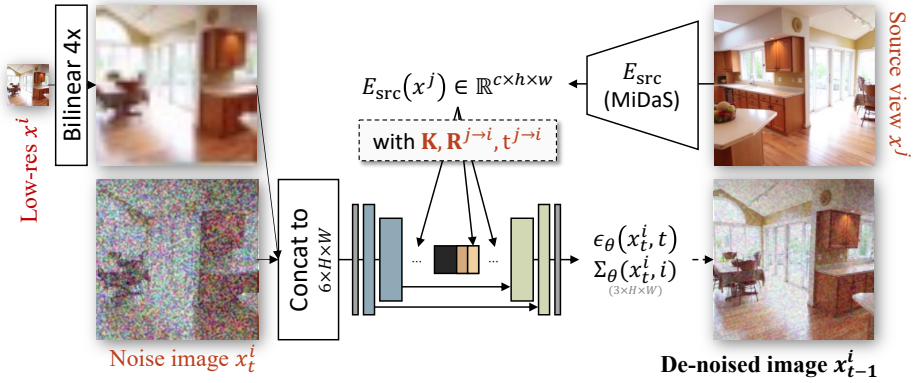

Figure 3. **Super-resolution pose-guided diffusion model.** We first bilinearly up-sample the low-resolution target view, and concatenate it with the noised high-resolution target view. Combining the features extracted from the high-resolution source view, we train the UNet network to de-noise the high-resolution target view.

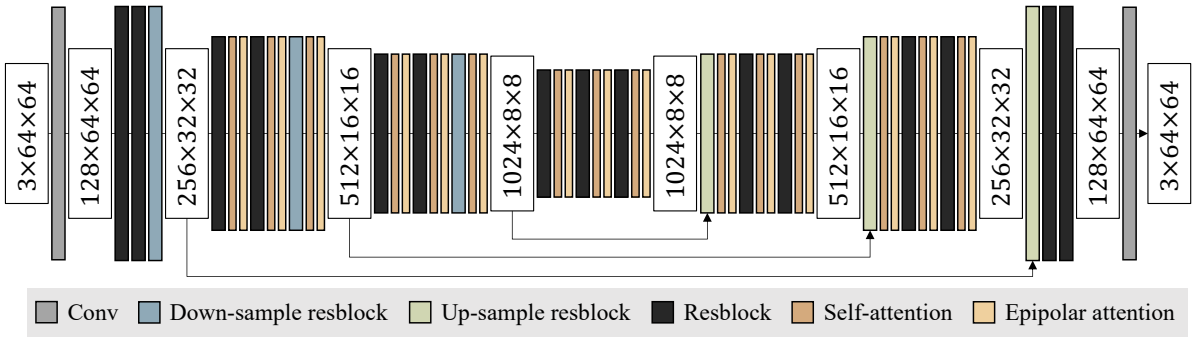

Figure 4. **UNet architecture.** We present the UNet architecture of the pose-guided diffusion model that produces 64×64 novel views.

Evaluation metrics. We use the AlexNet model to compute the LPIPS score, and InceptionV3 [11] network to calculate the FID/KID scores.[2][3][4] Both the AlexNet and InceptionV3 models are pre-trained on the ImageNet dataset. We use $30,000$ training images to pre-compute the real data statics for estimating the FID/KID scores. On the other hand, we use the pre-trained RAFT [12] model to compute the flow warping error E_{warp} [4].[5]

Reproducing results of previous methods. We describe how we evaluate the previous methods as follows:

- GeoGPT [10]: We use the official implementation.[6] For the evaluation conducted on the Re10K dataset, we use the

[2] https://github.com/richzhang/PerceptualSimilarity
[3] https://github.com/mseitzer/pytorch-fid
[4] https://github.com/GaParmar/clean-fid
[5] https://github.com/princeton-vl/RAFT
[6] https://github.com/CompVis/geometry-free-view-synthesis

Table 1. **Training details.** We provide the details of the UNet architecture and the training hyper-parameters.

Models	64	$64 \rightarrow 256$
Channels	128	128
Number of residual blocks	3	3
Channel multiples	$1, 2, 3, 4$	$1, 1, 2, 3, 4$
Head channels	64	64
Attention resolutions (self and epipolar)	$32, 16, 8$	16
Dropout (training)	0.1	0.1
Diffusion steps (training)	1000	1000
Noise schedule	cosine	cosine
Sampling steps (inference)	250	250
Sampling variance method	learned [6]	DDPM [2]
Batch size	64	32
Iterations	1M	1M
Learning rate	0.0001	0.0001
Adam β_2	0.999	0.999
Adam ϵ	1e-8	1e-8
EMA decay	0.9999	0.9999

pre-trained model parameters released on the Github webpage. Since the resolution of the generated images is slightly different from our setting, we center-crop and resize the images to 256×256 for evaluation. As for the MP3D dataset, we train the model using the default training parameters.

- LoR [9]: We use the official implementation to conduct the training and evaluation.[7] For the Re10K dataset, we train and evaluate the model using the default hyper-parameters. As for the MP3D dataset, we use the pre-trained model parameters provided by the authors to conduct the evaluation.

- SE3DS [3]: We use the official implementation and pre-trained model parameters to conduct the evaluation on the Re10K dataset.[8]

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. In *International Conference on 3D Vision*, 2017. 2

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 4

[3] Jing Yu Koh, Harsh Agrawal, Dhruv Batra, Richard Tucker, Austin Waters, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Simple and effective synthesis of indoor 3d scenes. *arXiv preprint arXiv:2204.02960*, 2022. 1, 4

[4] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *ECCV*, 2018. 3

[5] Zihang Lai, Sifei Liu, Alexei A Efros, and Xiaolong Wang. Video autoencoder: self-supervised disentanglement of static 3d structure and motion. In *ICCV*, 2021. 2

[6] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 4

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS workshop*, 2017. 2

[8] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *ECCV*, 2022. 1

[9] Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In *CVPR*, 2022. 1, 4

[10] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *ICCV*, 2021. 1, 3

[7] https://github.com/xrenaa/Look-Outside-Room

[8] https://github.com/google-research/se3ds

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 3

[12] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 3

[13] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM TOG*, 37(4):1–12, 2018. 2