GEN3C: 3D-Informed World-Consistent Video Generation with Precise Camera Control – Supplementary Material –

Xuanchi Ren* [1,2,3] Tianchang Shen* [1,2,3] Jiahui Huang[1] Huan Ling[1,2,3] Yifan Lu[1]
Merlin Nimier-David[1] Thomas Müller[1] Alexander Keller[1] Sanja Fidler[1,2,3] Jun Gao[1,2,3]
[1]NVIDIA [2]University of Toronto [3]Vector Institute

In the supplement, we provide additional details of our method in Sec. 1 and experiments in Sec. 2. We provide more qualitative results in Sec. 3.

1. Method Details

In this section, we provide additional details on the autoregressive generation process that updates the 3D cache.

1.1. Auto-regressive generation

In many applications, we need to generate a video with a sequence length that is longer than the length the original video models can support. To achieve this, we first divide the long video into overlapping chunks of length L, with a one-frame overlap between consecutive chunks, and generate the frames of each chunk sequentially in an autoregressive manner. Specifically, for the first chunk, we follow the inference pipeline described in Sec. 4.5 of the main paper to predict an RGB video. We then update the 3D cache with the frames from the first chunk prediction, which captures a new viewpoint of the scene and provides additional information not present in the original 3D cache.

To update the 3D cache, we estimate the pixel-wise depth of the last frame in the first chunk using DAV2 [10], and align this depth estimation with the 3D cache by minimizing the reprojection error. Specifically, we denote the depth estimation as $\mathbf{d}$ and optimize scaling s and translation t coefficients for $\mathbf{d}$. We render the point cloud from the 3D cache into a depth image at the camera viewpoint of $\mathbf{d}$. We render the point cloud from the 3D cache into a depth image from the camera viewpoint of $\mathbf{d}$, denoted as $\mathbf{d}^{\text{tgt}}$, and, similar to the main paper, render a mask M indicating whether each pixel is covered by the 3D cache. The optimization objective is then defined as:

$$s, t = \underset{s,t}{\operatorname{argmin}} \left\| \left(s \cdot \mathbf{d} + t - \mathbf{d}^{\text{tgt}}\right) \cdot M \right\|_2^2. \tag{1}$$

The optimized s and t are applied to normalize the depth estimation $\mathbf{d}$:

$$\mathbf{d}' = s \cdot \mathbf{d} + t. \tag{2}$$

We unproject $\mathbf{d}'$ into a 3D point cloud using the camera parameters of this frame and append it to the existing 3D cache. The updated 3D cache is subsequently used to predict the second chunk of frames. This ensures that the prediction for the second chunk is informed by the first chunk, leading to consistent generation of long videos. We iterate this process for all subsequent chunks.

2. Experimental Details

In this section, we provide additional details for our experiments in the main paper.

2.1. Optimization Details

We optimize the neural network using AdamW optimizer [6] with the learning rate 3e-5. During training, we apply a 15% dropout ratio to the conditions (the rendered videos from our 3D cache and the CLIP embedding of the first frame. We adopt a progressive training strategy for our model. Specifically, we first train the model on RE10K [12] and DL3DV [5] at a resolution of 320×576 for 100K iterations. We then finetune it on all four datasets at a higher resolution of 576×1024 for another 100K iterations. In the above two stages, the sequence length is set to 14. To support longer sequence lengths, we finetune the temporal layers of the video diffusion model on all four datasets for another 10K iterations at a resolution of 320×576. We first resize the video into the resolution of 576×1024 and use center cropping to get the 320×576 video. We randomly sample a video with a sequence length ranging from 14 to 56 frames to finetune the temporal layer. The entire training takes around 4 days using 32 A100 GPUs.

2.2. Inference Details

We follow the practices from Stable Video Diffusion [1] and use classifier-free guidance during inference with the guidance weight being 3. We run 25 diffusion steps to generate the result.

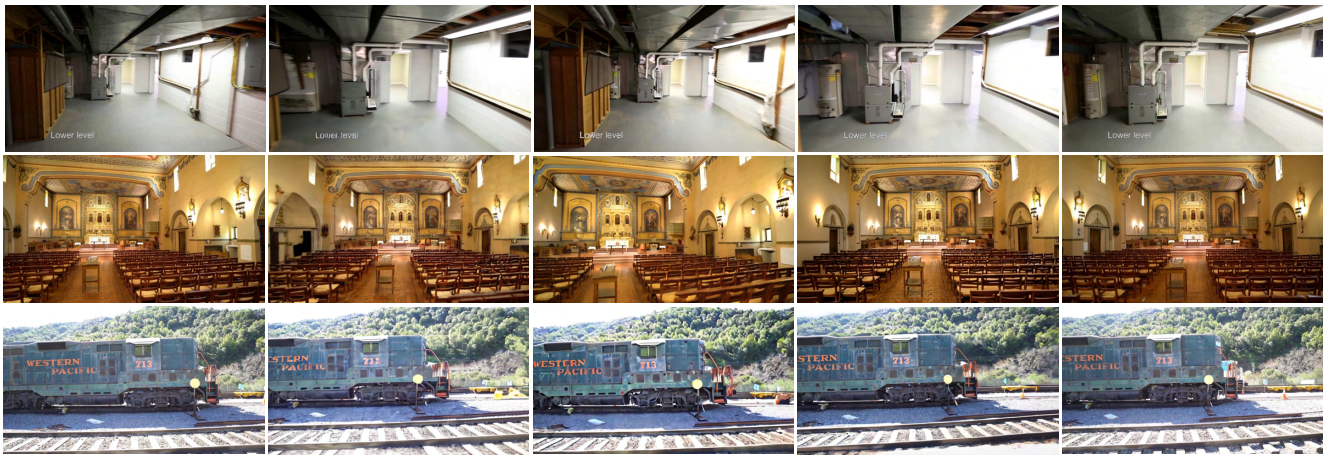

Figure 1. Additional qualitative results for single-view novel view synthesis.

Figure 2. Illustration of rendered depth images and model outputs. Our model can fix the error in the depth projection (such as the orange handrail in the first image and the light in the second one), and generate realistic content in the missing regions (such as the inpainted railway).

2.3. Single-view to video generation

We provide further details of the compared baselines in this subsection.

Baseline Details. For GenWarp [7][1] and MotionCtrl [9][2], we use the official checkpoint that is trained with Stable Video Diffusion [1] and evaluate on the same testing scenes as our method. Note that RE10k [12] is the training dataset for two methods. For NVS-Solver [11][3], we use the official codebase and run the evaluation using our testing data since the model is training-free.

[1] https://github.com/sony/genwarp
[2] https://github.com/TencentARC/MotionCtrl
[3] https://github.com/ZHU-Zhiyu/NVS_Solver

2.4. Two-views NVS

We provide further details of the compared baselines in this subsection.

Baseline Details. For PixelSplat [2], we take the official codebase and the released checkpoint[4] that is trained on RE10k [12] for a fair evaluation. For MVSplat [3], we also take the official codebase and the released checkpoint[5]. Both baselines take images and their corresponding camera parameters as input to reconstruct 3D Gaussian Splats that can be used to render the video in the target camera trajectory. We run the two baselines on the testing data we prepared and report the results for both interpolation and extrapolation.

2.5. NVS for driving simulation

Baseline Details. For Nerfacto, we take the official codebase released by Nerfstudio[6], which is a state-of-the-art codebase for training Nerf. For 3DGS, we take the official codebase[7]. We use all frames from three cameras in the training data to train Nerfacto and 3DGS.

Inference Pipeline. With a driving video, we estimate the depth of each frame and align it with the depth scale from the Lidar point cloud. We then unproject the depth estimation for each frame. In this case, we treat each frame as a different time capture of the same scene and concatenate them along the temporal dimension of the 3D cache, since there could exist dynamic objects in the scene. With the new camera trajectory provided by the user, we render the 3D cache along this trajectory. The rendering of the 3D

[4] https://github.com/dcharatan/pixelsplat
[5] https://github.com/donydchen/mvsplat
[6] https://github.com/nerfstudio-project/nerfstudio
[7] https://github.com/graphdeco-inria/gaussian-splatting

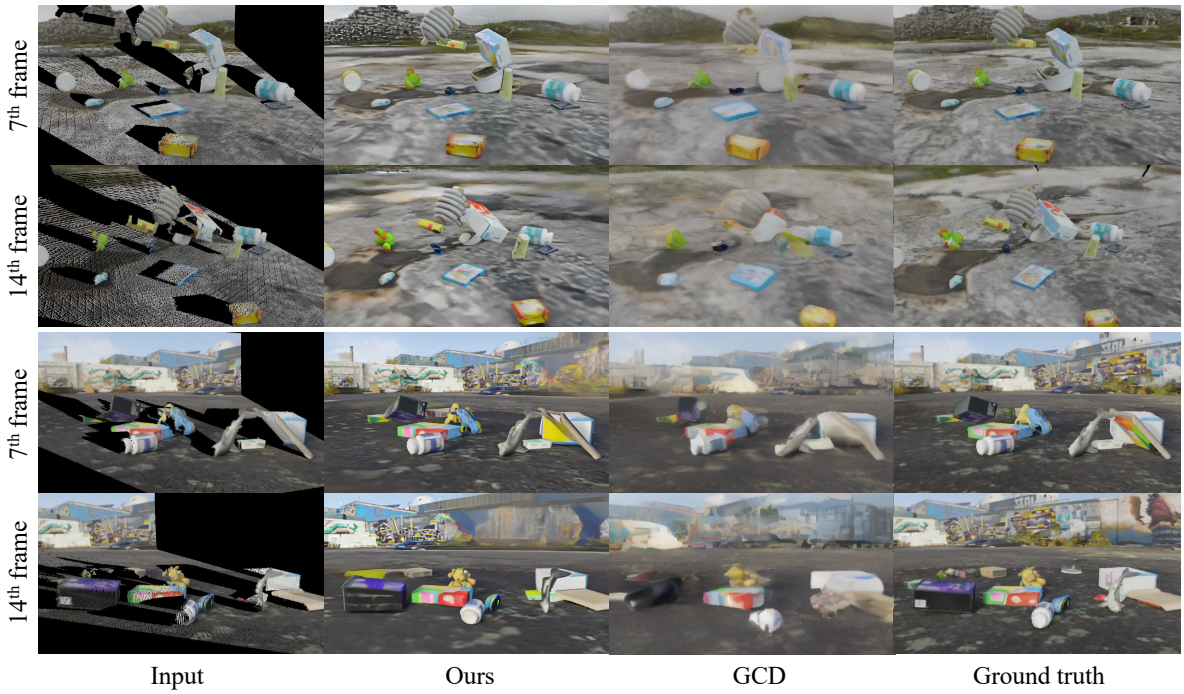

Figure 3. Qualitative results on Kubric4D [4]. Compared to GCD [8], our method demonstrates superior performance in preserving object details and dynamics in input videos.

Figure 4. Comparison of different strategies for incorporating masking information into the model. (Left) the mask channel is concatenated to the latent as an additional channel. (Right) the mask values are applied directly to the latent through element-wise multiplication.

cache is then used to generate the video at the novel camera trajectory.

2.6. Monocular Dynamic NVS

Inference Pipeline. With a monocular video of a dynamic scene, we aim to generate a video of the same scene from a different camera trajectory. Similar to the driving simulation, we predict the depth for each frame separately and concatenate the unprojected point cloud from depth along the temporal dimension of the 3D cache. With the new camera trajectory provided by the user, we render the 3D cache along this new camera trajectory. The rendered video is fed into GEN3C to generate a dynamic video output.

3. Additional Results

3.1. Generalization with mask channel

In Sec. 4.3 of the main paper, we discuss different strategies for incorporating mask information into the model. Here, we provide an additional qualitative comparison in Fig. 4. Concatenating the mask channel to the latent introduces additional model parameters, which do not generalize well to out-of-distribution masks during training. This issue is particularly severe in driving simulations, where ground truth views for novel trajectories (e.g., horizontal shifts from the original trajectory) are unavailable. In contrast, directly multiplying the mask with the latent demonstrates better generalization, significantly reducing artifacts when synthesizing extreme novel views.

3.2. Single-view to video generation

In addition to the comparison in the main paper, we provide the quantitative comparisons with GenWarp [7] and MotionCtrl [9] in Fig. 1. We also provide the rendered depth images and the model outputs in Fig. 2 to demonstrate the capability of our model on both filling missing regions in the 3D cache and fixing artifacts from the rendering of the imperfect 3D cache.

3.3. Monocular Dynamic Novel View Synthesis

We provide the qualitative comparison with GCD [8] on the Kubric dataset in Fig. 3. Our method generates sharper video details with more object details and dynamics compared to GCD [8].

References

[1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 1, 2

[2] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. PixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *CVPR*, 2024. 2

[3] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: efficient 3D Gaussian splatting from sparse multi-view images. In *ECCV*, 2024. 2

[4] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3749–3761, 2022. 3

[5] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. DL3DV-10K: a large-scale scene dataset for deep learning-based 3D vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024. 1

[6] Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017. 1

[7] Junyoung Seo, Kazumi Fukuda, Takashi Shibuya, Takuya Narihira, Naoki Murata, Shoukang Hu, Chieh-Hsin Lai, Seungryong Kim, and Yuki Mitsufuji. Genwarp: Single image to novel views with semantic-preserving generative warping. In *NeurIPS*, 2024. 2, 3

[8] Basile Van Hoorick, Rundi Wu, Ege Ozguroglu, Kyle Sargent, Ruoshi Liu, Pavel Tokmakov, Achal Dave, Changxi Zheng, and Carl Vondrick. Generative camera dolly: Extreme monocular dynamic novel view synthesis. *European Conference on Computer Vision (ECCV)*, 2024. 3, 4

[9] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *SIGGRAPH*, 2024. 2, 3

[10] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. *arXiv:2406.09414*, 2024. 1

[11] Meng You, Zhiyu Zhu, Hui Liu, and Junhui Hou. NVS-Solver: video diffusion model as zero-shot novel view synthesizer. *CoRR*, abs/2405.15364, 2024. 2

[12] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 1, 2