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

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

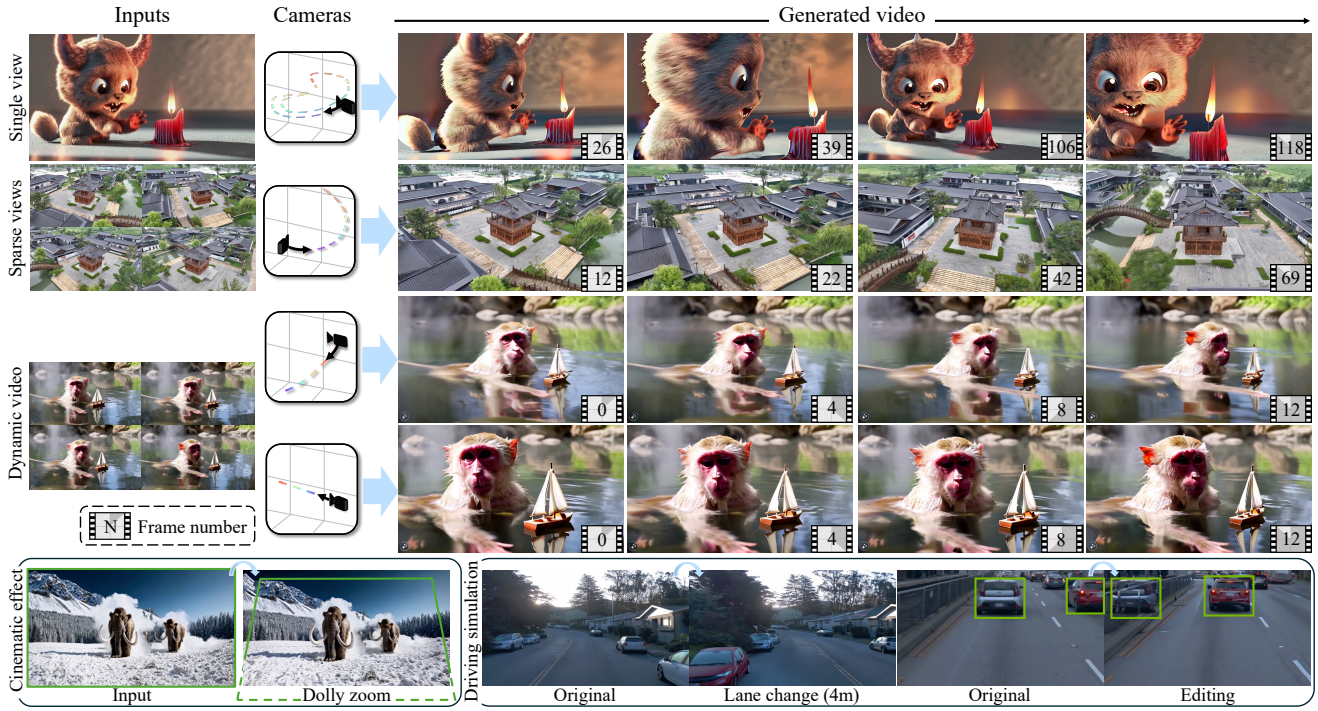


Figure 1. GEN3C can generate long and temporally consistent videos with precise camera control. We apply it to various applications, including single-view and sparse-views novel view synthesis, monocular dynamic video novel view synthesis, and driving simulation. With an explicit 3D cache, GEN3C further supports generating videos with cinematic effects, such as Dolly Zoom which simultaneously changes poses and intrinsics, and 3D editing.

Abstract

We present GEN3C*, a generative video model with precise* **C***amera* **C***ontrol and temporal 3D* **C***onsistency. Prior video models already generate realistic videos, but they tend to leverage little 3D information, leading to inconsistencies, such as objects popping in and out of existence. Camera control, if implemented at all, is imprecise, because camera parameters are mere inputs to the neural network which must then infer how the video depends on the camera. In contrast,* GEN3C *is guided by a 3D cache: point clouds obtained by predicting the pixel-wise depth of seed images or previously generated frames. When generating the next frames,* GEN3C *is conditioned on the 2D renderings of the 3D cache with the new camera trajectory provided by the user. Crucially, this means that* GEN3C *neither has to remember what it previously generated nor does it have to infer the image structure from the camera pose. The model, instead, can focus all its generative power on previously unobserved regions, as well as advancing the scene state to the next frame. Our results demonstrate more precise camera control than prior work, as well as state-of-the-art results in sparse-view novel view synthesis, even in challenging settings such as driving scenes and monocular dynamic video. Results are best viewed in videos. Check out our webpage!* https://research.nvidia.com/labs/toronto-ai/GEN3C/

1. Introduction

Creating immersive visual renderings that convey real-world scenery while enabling flexible viewing, manipulation and simulation thereof, is a longstanding aspiration in computer graphics, supporting industries including movie production,

VR/AR, robotics and social platforms. However, traditional graphics workflows entail extensive manual effort and time in asset creation and scene design. Recently, Novel View Synthesis (NVS) methods [25, 36] unleash this requirement and successfully produce realistic images at novel viewpoints of a scene with a set of posed images. However, such methods generally require dense input images and often suffer from severe artifacts when viewing from extreme viewpoints.

More recently, video generation models, which can "render" photorealistic videos from text prompts, have demonstrated impressive visual quality and powerful content creation capabilities [4, 35, 40, 45], capturing the underlying distribution of real-world videos by training with massive amounts of data. However, the key challenge towards practical applications in digital content creation workflows is controllability and consistency, *i.e*. allowing the user to adjust camera motion, scene composition and dynamics, and maintaining spatial and temporal consistency across long-generated videos. While several methods have been proposed to address this challenge through fine-tuning with images, additional text prompts or camera parameters [16, 31, 48, 54, 60, 61], achieving precise control for subtle or complex camera movements or scene arrangement remains unsolved. The model can easily forget previously generated content when looking back and forth; see Fig. 2.

Controllability and consistency in graphics pipelines are fundamentally rooted in their explicit modeling of 3D geometry and rendering it into 2D views. In this work, we take an initial step towards building this insight into the video generation models, and propose GEN3C, a world-**C**onsistent video generation model with precise **C**amera **C**ontrol. Its core is an approximated 3D geometry—akin to 3D modeling in graphics pipelines—constructed from user-provided images, and can be precisely projected to any camera trajectory to guide video generation, providing strong conditioning for visual consistency. In addition, "rendering" with video generation models leverages the rich prior from pre-trained large models, enabling NVS in sparse-view settings.

Specifically, we construct a 3D cache, represented as a point cloud, by unprojecting a depth estimate of the input image(s) or previously generated video frames. With the camera trajectory from the user, we then render the 3D cache and use the rendered video as conditioning input for the video model. The video model is fine-tuned to translate imperfectly rendered video into a high-quality video, correcting any artifacts that stem from the 3D unprojection-projection process and filling in missing information. This way, we achieve precise control of the camera and encourage the generated video to remain consistent over time. When multiple views are provided, we maintain a separate 3D cache for each individual view and leverage the video model to handle potential misalignment and aggregattion across views. Acting as an explicit geometry, the 3D cache further enables scene manipulation by simply modifying the 3D point cloud.

Figure 2. **Motivation:** Our model can generate consistent videos when the camera covers the same region multiple times, while previous work produces severe artifacts due to the lack of explicit modeling of the history.

We extensively evaluate our model on video generation tasks with varying input conditions, ranging from a single image to sparse and dense multi-view inputs. Our model generalizes well to dynamic scenes and demonstrates the ability to accurately control viewpoint, generate 3D-consistent high-fidelity videos, and fill in occluded or missing regions in the 3D cache. Beyond novel view synthesis, we explore applications enabled by the explicit 3D cache, such as object removal, and scene editing. We believe these results validate our approach as a step toward applying video generation models in production and simulation environments.

2. Related Work

Novel View Synthesis (NVS). Generating novel views from a set of posed images has seen significant progress [25, 36, 51], with numerous extensions towards large scene reconstruction [3, 28, 63, 72], improved rendering quality [2, 21, 59], faster rendering speed [38, 59] and handling dynamic scenes [11, 33]. Yet, many of these methods require a dense set of input images and may produce severe artifacts when viewed from extreme viewpoints. Several works proposed to address these issues through regularization using geometric priors [10, 39, 44, 49, 55, 64, 71, 76], which, however, are sensitive to noise in the estimated depth or normals. Alternative approaches seek to train a feed-forward model to predict novel views from sparse posed images [6–8, 22, 43, 52, 56, 68, 74], but these methods are limited by the scarcity of training data and struggle to generalize to unseen domains and extreme novel views. With the recent success of image/video generation models, ReconFusion [62] and CAT3D [12] started leveraging prior knowledge learned by these models to facilitate sparse-view NVS. Due to the necessity of per-scene optimization, these methods remain inherently slow. Concurrent and unpublished work, including ReconX [30] and ViewCrafter [70], are closer to our work. However, they rely on the alignment of input multiple views using DUSt3R [57], which is not robust to thin structures, and introduces artifacts when misalignment happens. MultiDiff [37] leverages depth to warp a single image as guidance for novel view synthesis using a video diffusion model. It, however, only focuses on single view setting.

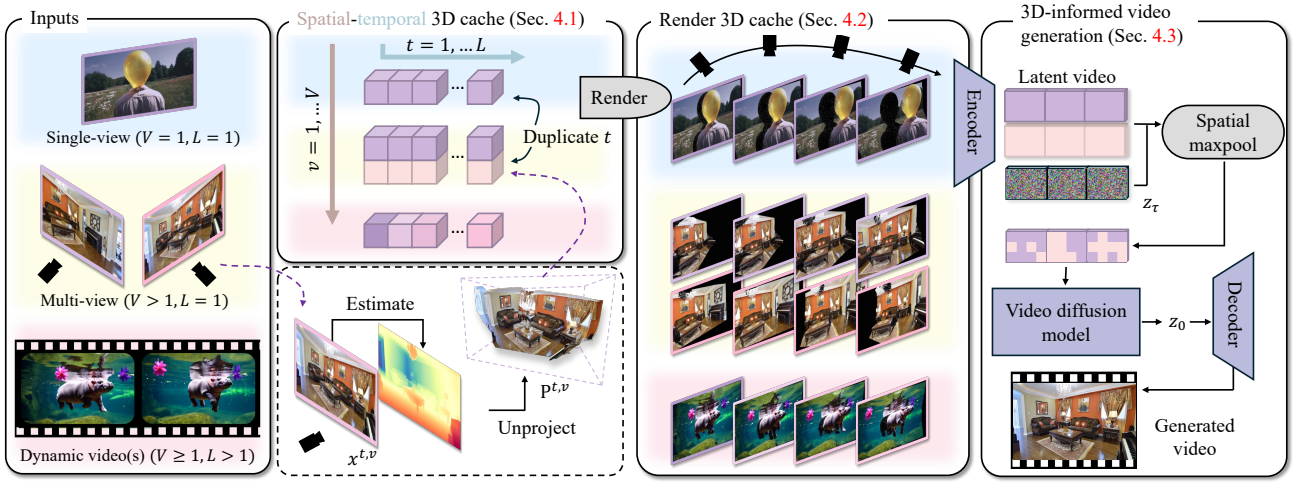


Figure 3. Overview of GEN3C. With the user input, which can be a single-view image, multi-view images, or dynamic video(s), we first build a spatiotemporal 3D cache (Sec. 4.1) by predicting the depth for each image and unprojecting it into 3D. With the camera poses from the user, we then render the cache into video(s) (Sec. 4.2), which are fed into the video diffusion model to generate a photorealistic video that aligns with the desired camera poses (Sec. 4.3 & 4.4).

Camera-Controllable Video Generation. Early works propose inputting numerical camera parameters into their video generation models as an additional condition to fine-tune for camera control [15, 31, 48, 54, 60, 61]. However, these works struggle with precise control due to the model having to learn the mapping from the camera parameters to video, usually failing to generalize to camera motion that is different from the training data. Several training-free methods [19, 67] proposed to leverage depth to warp a single frame to a given camera trajectory and incorporate the result in the denoising process of a pre-trained diffusion model. This requires tuning the degree of consistency between the depth-warped images and the denoising output, leading to either artifacts or imprecise camera control.

Consistent Video Generation. Early work [34] leverages a 3D point cloud, similar to our 3D cache, obtained by applying structure from motion to past frames. Renders of this point cloud are used to condition a Generative Adversarial Network [13]. Instead, we estimate the depth for each seed image that is then reconciled by a diffusion-based video generation model, which is more robust to small-overlap images. Streetscapes [9] improved the consistency of video diffusion models by relying on a precise height map of the environment that is not necessarily available. More recently, CVD [27] make synchronous frames of generated videos consistent with each other. However, overall consistency is still lost if content temporarily leaves the view of *all* videos, because no history is maintained. StreamingT2V [17] maintain a history in the form of latent feature maps to enhance consistency, but camera control remains difficult because the history is latent rather than 3D.

3. Background: Video Diffusion Models

As our method is based on a video diffusion model, we briefly review their principles. A diffusion model $\mathbf{f}_\theta$ learns to model a data distribution $p_{\text{data}}(\mathbf{x})$ via an iterative denoising process. To train the model, noisy versions $\mathbf{x}_\tau = \alpha_\tau \mathbf{x}_0 + \sigma_\tau \epsilon$ of a data sample $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$ are constructed by adding noise ϵ sampled from a Gaussian distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$, with the noise schedule parameterized by α_τ and σ_τ. The diffusion time τ is sampled from the distribution p_τ. Then, the parameters θ of the diffusion model $\mathbf{f}_\theta$ are optimized to minimize the denoising score matching objective function:

$$\mathbb{E}_{\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}), \tau \sim p_\tau, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\| \mathbf{f}_\theta(\mathbf{x}_\tau; \mathbf{c}, \tau) - \mathbf{y} \|_2^2 \right], \quad (1)$$

where $\mathbf{c}$ is optional conditions, and the target $\mathbf{y}$ can be ϵ, $\alpha_\tau \epsilon - \sigma_\tau \mathbf{x}_0$ [46], or $\mathbf{x}_0$ [23], depending on the selected denoising process. Once trained, iteratively applying $\mathbf{f}_\theta$ to a sample of Gaussian noise will produce a sample of $p_{\text{data}}(\mathbf{x})$.

In diffusion-based video generation models, latent diffusion models [5] are frequently employed to compress the video for operation in a lower-dimensional space. Specifically, given the a RGB video data $\mathbf{x} \in \mathbb{R}^{L \times 3 \times H \times W}$, where L is the number of frames of size $H \times W$, a pre-trained VAE encoder $\mathcal{E}$ will encode the video into a latent space, i.e. $\mathbf{z} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{L' \times C \times h \times w}$. Training and inference of the diffusion model are performed in this latent space. The final video $\hat{\mathbf{x}} = \mathcal{D}(\mathbf{z})$ is decoded with a pre-trained VAE decoder $\mathcal{D}$. In this paper, we leverage the pre-trained Stable Video Diffusion [4] model, which is conditioned on an image $\mathbf{c}$ and only compresses the video in the spatial dimension: $L' = L, C = 4, h = \frac{H}{8}$, and $w = \frac{W}{8}$. However, our method is compatible with any other image-to-video diffusion model, as it does not rely on details of its architecture.

4. Method: 3D-Informed Video Generation

Our key idea is to use 3D guidance to inform video generation, enabling precise camera control and improving consistency across the video frames. For this purpose, we first build a 3D cache from the input image(s) or pre-generated

video frames (Sec. 4.1). Then, the 3D cache will be rendered into the camera plane with the camera poses from the user (Sec. 4.2). Such renderings, while imperfect, provide a strong condition for the video generation model on the visual content it needs to generate (Sec. 4.3). Our video generation model is fine-tuned accordingly to produce a 3D-consistent video that precisely aligns with the desired camera poses (Sec. 4.4). Fig. 3 provides an overview of our method.

4.1. Building a Spatiotemporal 3D Cache

Choosing a proper 3D cache compatible with different applications and that generalizes to different scenes is the main consideration in our design. Recently, depth estimation has achieved significant progress across various domains [20, 24, 57, 65, 66], such as indoor, outdoor, or self-driving scenes. We thus choose the colored point cloud, unprojected from the depth estimation of an RGB image, as the basic element of our 3D cache.

Specifically, we maintain a spatiotemporal 3D cache. For an RGB image seen from a camera viewpoint v at time t, we create a point cloud, $\mathbf{P}^{t,v}$, by unprojecting the depth estimation of this RGB image. We denote the number of camera views as V, and the length in temporal dimension as L; thus, our 3D cache is an $L \times V$ array of point clouds.

We build the spatiotemporal 3D cache according to the specific downstream applications. For *single image to video generation*, we only create one cache element ($V = 1$) for the given image, and duplicate it L times along the temporal dimension to generate the video of length L. For *static NVS*, we create one cache element for each of the V images provided by the user, and duplicate them L times along the temporal dimension. We can then enable both sparse-view and dense-view NVS. For *dynamic NVS*, we build the cache from each initial video(s) of identical length L that are provided by a user or generated by another video model. Then, V equals to the number of time-synchronized videos, and we can enable both single-view and multi-view dynamic NVS. For these different applications, we assume the camera poses are provided along with the image(s) or video(s). If not, we estimate the camera poses using DROID-SLAM [53].

Optionally, the 3D cache we build may be edited or simulated, for example, by removing or adding objects; we provide qualitative results in Sec. 5.4.

4.2. Rendering the 3D Cache

Point clouds can be easily and efficiently rendered along any camera trajectory, much like Gaussian Splats [25]. Such a rendering function $\mathcal{R}$ maps $\mathbf{P}^{t,v}$ onto a tuple: $(I^{t,v}, M^{t,v}) := \mathcal{R}(\mathbf{P}^{t,v}, \mathbf{C}^t)$, where $I^{t,v}$ is the RGB image as seen from a new camera $\mathbf{C}^t$. The mask $M^{t,v}$ identifies disocclusions, flagging pixels that are not covered when rendering the point cloud. In that sense, the mask identifies regions of the image $I^{t,v}$ that need to be filled in.

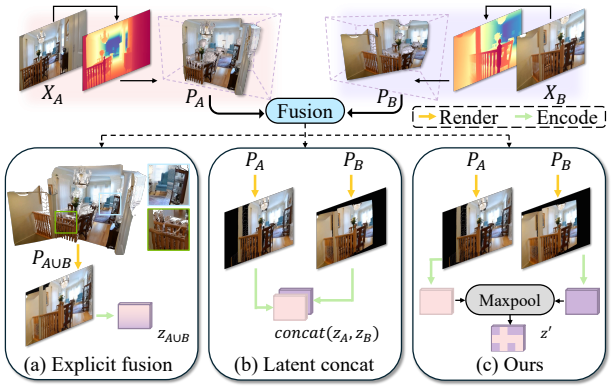


Figure 4. Three approaches to fuse the point cloud from two views.

For a sequence $\mathbf{C} = (\mathbf{C}^1, \ldots, \mathbf{C}^L)$ of new camera poses from the user, we render all cache elements $\mathbf{P}^{t,v}$ and obtain V videos $(\mathcal{R}(\mathbf{P}^{1,v}, \mathbf{C}^1), \mathcal{R}(\mathbf{P}^{2,v}, \mathbf{C}^2), \ldots, \mathcal{R}(\mathbf{P}^{L,v}, \mathbf{C}^L))$, where $1 \leq v \leq V$. Concatenating the rendered images $(I^{1,v}, \ldots, I^{L,v})$ and masks $(M^{1,v}, \ldots, M^{L,v})$ along the temporal dimension for a camera view v, we denote the resulting videos of the images and the masks by $\mathbf{I}^v \in \mathbb{R}^{L\times 3\times H\times W}$ and $\mathbf{M}^v \in \mathbb{R}^{L\times 1\times H\times W}$, respectively.

4.3. Fusing and Injecting the 3D Cache

When conditioning a video diffusion model with the renderings of our 3D cache, the key challenge is that the 3D cache may be inconsistent across different camera viewpoints, either due to imperfect depth predictions or inconsistent lighting. Hence, the model will need to aggregate the information (if $V > 1$) for a coherent prediction. Our key principle when designing this module is to minimize the introduction of additional trainable parameters: as the pre-trained video diffusion model has been trained on massive Internet data, any new parameters may not generalize as well.

Specifically, we modify the forward computation process of the image-to-video diffusion model, denoted by $\mathbf{f}'_\theta$. We first encode the rendered video $\mathbf{I}^v$ using the frozen VAE encoder $\mathcal{E}$ to obtain the latent video $\mathbf{z}^v = \mathcal{E}(\mathbf{I}^v)$, and mask out the regions that are not covered by the 3D cache, as indicated by the masks $\mathbf{M}^v$. During training, we then concatenate the masked latent with the noisy version $\mathbf{z}_\tau = \alpha_\tau \mathcal{E}(\mathbf{x}) + \sigma_\tau \epsilon$ of the target video $\mathbf{x}$ in latent space along the channel dimension, and feed it into the video diffusion model. To fuse the information from multiple viewpoints, we separately feed each viewpoint into the first layer of the diffusion model, denoted by *In-Layer*, and apply max-pooling over all the viewpoints to get the final feature map. In summary:

$$\mathbf{z}^{v,\prime} = \textit{In-Layer}(\text{Concat}(\mathbf{z}^v \odot \mathbf{M}^{v,\prime}, \mathbf{z}_\tau)), \quad (2)$$

$$\mathbf{z}' = \text{Max-Pool}\{\mathbf{z}^{1,\prime}, \ldots, \mathbf{z}^{V,\prime}\}, \quad (3)$$

where $\odot$ denotes element-wise multiplication and $\mathbf{M}^{v,\prime} \in \mathbb{R}^{L\times 1\times h\times w}$ is obtained by downsampling the $\mathbf{M}^v$ using min-pooling with size $\frac{H}{h} \times \frac{W}{w}$, in order to align with the latent

Figure 5. Qualitative results for single-view novel view synthesis. Compared to the baselines, our model generates photorealistic novel view images that precisely align with the camera poses. The green zoom-in boxes highlight the fine-grained details that our model can preserve.

dimension (Sec. 3). The resulting feature map $\mathbf{z}'$ is further processed in the video diffusion model that is fine-tuned to generate a consistent video conditioned on these renderings from the 3D cache.

Discussion. The strategy described above is a general mechanism to aggregate the information from multiple views and inject it into the video diffusion model. We compare it to alternatives that exhibit different properties, as illustrated in Fig. 4. See Sec. 5.6 for an empirical comparison.

The **explicit fusion** approach, proposed in concurrent works [30, 70], directly fuses the point clouds in 3D space. While being simple, such an approach strongly relies on depth alignment and will introduce artifacts when inconsistencies manifest across multiple viewpoints. Furthermore, it would be nontrivial to imbue such fused cache with view-dependent lighting information. For these reasons, we prefer to let the model handle aggregation of view information. Another approach, which we denote as "**concat**", is to concatenate all latents for the rendered cache along the channel dimension. Although this approach empirically works well, it requires bounding the maximum number of viewpoints the model can support by a constant, and imposes an order on the viewpoints. Instead, we favored a permutation-invariant fusion operation, leading to our pooling-based strategy.

Another key design choice is the incorporation of the masking information into the model. We initially tried to concatenate the mask channel to the latent. However, the concatenation operation introduces additional model parameters, which would now need to be trained, and therefore may not generalize well when the mask channel is not represented in any large-scale training data. Instead, we apply the mask values directly to the latents by element-wise multiplication, leaving the model architecture unchanged. We provide emperical results of this observation in the Supplement.

4.4. Model Training

With the renderings of the 3D cache as the conditioning signal $\mathbf{c}$, we fine-tune our modified video diffusion model $\mathbf{f}'_\theta$. Specifically, we first create pairs of the renderings of the 3D cache, $\mathcal{R}(\mathbf{P}^{t,v}, \cdot)$, and the corresponding RGB ground truth video $\mathbf{x}$ along a new camera trajectory from our training data. We then fine-tune the video diffusion model using our fusion strategy (Sec. 4.3) with the denoising score matching objective function of Eqn. (1), where the target $\mathbf{y}$ is $\mathbf{z}_0 = \mathcal{E}(\mathbf{x})$ following the pre-trained image-to-video diffusion model practices [4]. We also encode the first frame using the CLIP model [41] as an additional condition. Details about creating paired data used for fine-tuning are provided in Sec. 5.

4.5. Model Inference

For inference, we initialize the latent code $\mathbf{z}$ with Gaussian noise and iteratively denoise this latent code using our modified video diffusion model $\mathbf{f}'_\theta$, conditioned on the renderings of our 3D cache. The final RGB video is obtained by running the pre-trained VAE decoder $\mathcal{D}$ on the denoised latent code. Generating a 14-frame video takes around 30 seconds on one A100 NVIDIA GPU.

Autoregressive Inference and 3D Cache Updates. Many applications require generating long videos, but, the longer the video, existing models are particularly prone to inconsistencies. To generate long, *consistent* videos, we propose updating our 3D cache incrementally. We first divide the long video into overlapping chunks of length L with a one-frame overlap between two consecutive chunks. We then render the 3D cache and generate the frames of each chunk in sequence autoregressively. To make the prediction consistent over time, we update the 3D cache using previously generated chunks: for each generated frame in a chunk, we estimate its pixel-wise depth using a depth estimator [66]. Since the camera pose of the frames is known (user-provided), we can align the depth estimation with the existing 3D cache by minimizing reprojection error; details in the supplement.

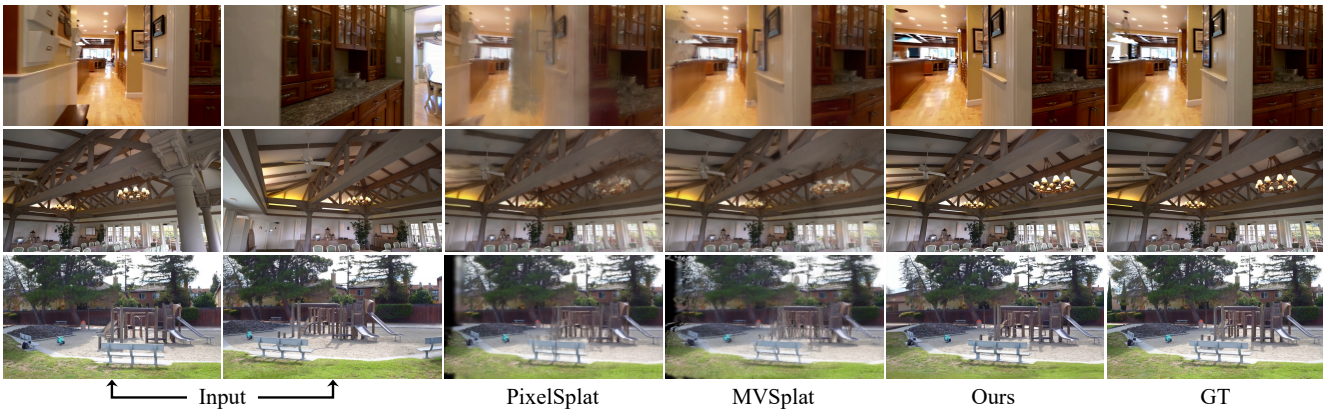


Figure 6. Qualitative results on two-views novel view synthesis. Compared to the baselines, our model generates much more plausible and realistic novel views with a smooth transition between two input views even if the overlap is small (such as the first row).

The aligned RGB-D frame is then unprojected into 3D and concatenated with the 3D cache, which is subsequently used for predicting the next chunk of frames.

5. Experiments and Applications

In this section, we introduce the experimental setup to train GEN3C (Sec. 5.1) and showcase its versatility with several downstream tasks, including single image to video generation (Sec. 5.2), two-views NVS (Sec. 5.3), NVS for driving simulation (Sec. 5.4), and monocular dynamic NVS (Sec. 5.5). We provide ablation studies in Sec. 5.6.

5.1. Training Details

A key challenge in training GEN3C is the lack of multi-view, dynamic, real-world video data, that provides the pairs of 3D cache and ground truth video for a novel camera trajectory. We leverage static real-world video to help the model reason about spatial consistency and synthetic multi-view dynamic video to help with temporal consistency.

Datasets. We select three real-world videos datasets: RE10K [75], DL3DV [29], Waymo Open Dataset (WOD) [50], and a synthetic dataset Kubric4D [14, 54]. RE10K [75] consists of 74,766 video clips that capture real-world real-estate scenes for both indoors and outdoors. We estimate camera parameters with DROID-SLAM [53] and predict per-frame depth using DAV2 [66]. The depth prediction is aligned with the scene scale from DROID-SLAM. DL3DV [29] contains 10k videos of real-world scenes. We annotate these clips following the same protocol as RE10K. WOD [50] is a real-world driving dataset with 1000 scenes and each scene has 200 frames. We use DAV2 [66] to predict the depth and rigidly align it with the scale from the LiDAR point cloud. For Kubric4D [14], we use the 3000 scenes generated by GCD [54], which includes multi-object dynamics. This dataset is in the format of point cloud sequences and we render RGB-D videos for desirable camera trajectories.

Paired Data Curation. For real-world videos, we train our model to predict current frames using past or future frames from the same sequence. In this way, we effectively simulate viewpoint changes, allowing the model to extrapolate to unseen viewpoints and generate consistent video informed by the observations. Specifically, for RE10K and DL3DV, we randomly select equally-spaced $V \in [1, 4]$ frames from the video clip to create our 3D cache. The ground truth video consists of L consecutive frames that include one of the selected V frames. For the WOD, we randomly sample a sequence of time-synchronized L frames for each of the three cameras to create our 3D cache ($V = 3$ in this case). The ground truth video is only sampled from the FRONT camera. We use all three cameras to create the 3D cache because it allows the model to learn to resolve inconsistencies across cameras, such as depth prediction, camera ISP, etc. For Kubric, we render each dynamic scene from $V \in [1, 4]$ camera trajectories to generate multi-view dynamic videos for creating the 3D cache. We then render one additional video at a different camera trajectory as ground truth. Although we choose maximum V to be 4 for training, it is flexible and can be an arbitrary number.

	Tanks-and-Temples [26]			RE10K [75]			
Method	PSNR ↑	SSIM ↑	LPIPS↓	PSNR ↑	SSIM ↑	LPIPS↓	TSED↑
MotionCtrl [60]	13.46	0.46	0.42	13.60	0.59	0.46	0.1363
CameraCtrl [16]	15.88	0.55	0.29	18.40	0.72	0.25	0.8033
GenWarp [47]	16.04	0.50	0.39	15.50	0.61	0.40	0.0330
NVS-Solver [67]	16.95	0.59	0.27	16.90	0.69	0.30	0.7286
GEN3C	**18.66**	**0.67**	**0.20**	**19.88**	**0.78**	**0.20**	**0.9143**

Table 1. Quantitative results for single view to video generation. *RE10K* is the in-domain dataset and *Tanks and Temples* is the out-of-domain dataset.

5.2. Single View to Video Generation

GEN3C can be easily applied to video/scene creation from a single image. We first predict the depth of the given image, then create the 3D cache, and render it into a 2D video, which is fed into the trained video diffusion model to generate a video that precisely follows the given camera trajectory.

Evaluation and Baselines. We compare GEN3C to four baselines, including GenWarp [47], MotionCtrl [60], CameraCtrl [16], and NVS-Solver [67]. For a fair comparison to GenWarp [47] and NVS-Solver [67], we use the same depth estimator [66] to get the pixel-wise depth and rigidly align it by globally shifting and scaling using the scene scale. CameraCtrl [16] is the most related work, and we reproduce it by training with the same datasets, training protocol, and video

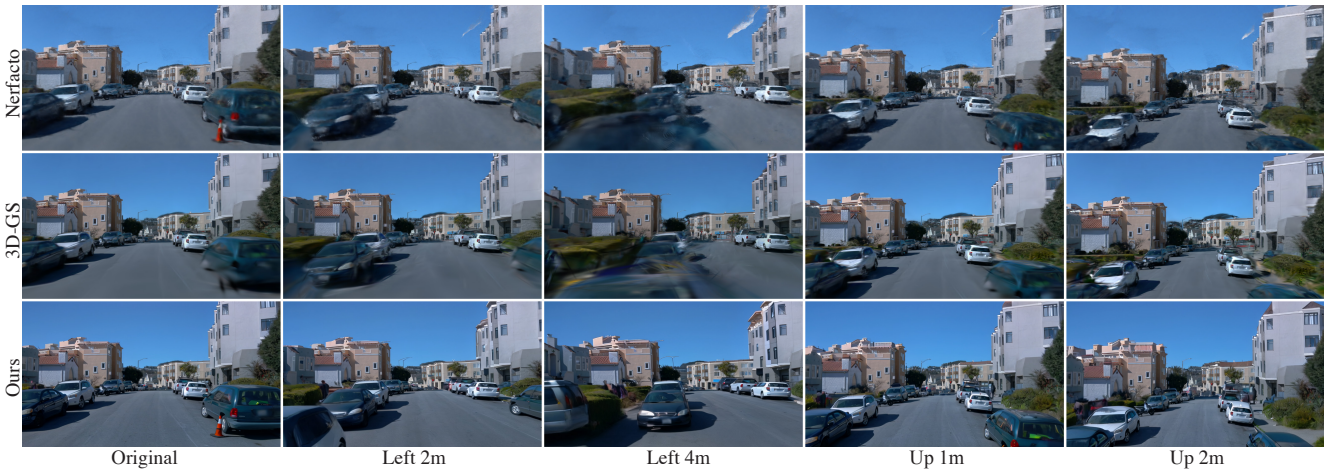


Figure 7. Qualitative results on novel view synthesis for driving scene. Our model can fill in the missing regions in the original video even when the deviation is large, while reconstruction-based baselines produce severe artifacts.

	Method	PSNR ↑	SSIM ↑	LPIPS↓
T-&-T	PixelSplat [6]	21.34 / 17.45	0.70 / 0.65	0.42 / 0.46
	MVSplat [8]	20.90 / 16.08	0.70 / 0.63	0.39 / 0.44
	GEN3C	**22.22 / 20.51**	**0.76 / 0.72**	**0.14 / 0.16**
RE10K	PixelSplat [6]	19.74 / 16.95	0.75 / 0.70	0.40 / 0.44
	MVSplat [8]	21.40 / 15.51	0.78 / 0.69	0.30 / 0.37
	GEN3C	**24.08 / 21.56**	**0.86 / 0.83**	**0.11 / 0.15**

Table 2. Quantitative results for two-views NVS. The two values in each table cell represent the interpolation and extrapolation results, respectively.

diffusion model as in our method and replacing the rendered videos from our 3D cache with Plücker embeddings of camera trajectories.We evaluate all the methods on two datasets: *RE10K*, which serves for in-domain testing. and *Tanks and Temples (T-&-T)*, which serves for out-of-domain testing to evaluate generalization capabilities. To ensure a comprehensive evaluation, we sample 100 testing sequences for both *RE10K* and *T-&-T*. Following prior work [8, 42, 70], we report both pixel-align metrics, *i.e.*, PSNR and SSIM [58], and perceptual metrics, *i.e.*, LPIPS [73]. We further report TSED score [69] to evaluate the 3D consistency of the prediction.

Results. Quantitative results are provided in Table 1. Our method outperforms all the baselines in both out-of-domain and in-domain testing, demonstrating the strong capability of generating photorealistic videos from a single image. Notably, Plücker-embedding based methods, such as CameraCtrl [16], generalize poorly to out-of-domain data, which has both different scene layouts and camera trajectories. Thanks to the explicit modeling of 3D content in our 3D cache, our model only suffers a small performance drop. We provide a qualitative comparison with the two strongest baselines in Fig. 5 and with the others in the Supplement. The prediction from our method precisely follows the ground truth camera trajectory and captures fine-grained detail such as the chair legs or letter words. In particular, CameraCtrl [16] fails to precisely follow the camera motion, as reasoning the scene layout only from the Plücker embedding is hard.

Method	$y \pm 0.0$m	$y \pm 1.0$m	$y \pm 2.0$m	$y \pm 4.0$m
Nerfacto [51]	48.34	67.77	80.41	112.40
3D-GS [25]	34.81	53.85	61.78	81.26
GEN3C	**7.93**	**18.19**	**25.11**	**35.33**

Table 3. Quantitative results of FID [18] for NVS on driving scene. GEN3C significantly outperforms the baselines, especially when generating novel views that are far away from the original trajectory.

5.3. Two-Views Novel View Synthesis

We further apply GEN3C to a challenging sparse-view novel view synthesis setting, where only two views are provided and we generate novel views from these two views. Similar to Sec. 5.2, we first predict the depth for each view, create the 3D cache, and use the camera trajectory to render it into two videos, which are fed into and fused by GEN3C to generate the output video. Note that during inference, our model is not limited to two views and can be applied to any number of views. We provide qualitative results in the Supplement.

Evaluation and Baselines. We compare our method to two representative works for sparse view reconstruction, PixelSplat [6] and MVSplat [8]. In this task, we evaluate both the interpolation and extrapolation capabilities of our model. Specifically, we randomly select two input frames from a video. For interpolation, we select target views between the input frames, and for extrapolation, we choose target views outside the range of the two input frames. We sample 40 testing sequences from both *RE10K* [75] and *T-&-T* [26], and report PSNR, SSIM and LPIPS.

Results. We provide quantitative results in Table 2 with qualitative results in Fig. 6. Our method outperforms all the baselines, especially when extrapolating from the provided two views, and can generate photorealistic novel views even if the overlap between two views is small, thanks to the strong prior from the pre-trained video generation models.

5.4. Novel View Synthesis for Driving Simulation

Simulating real-world driving scenes along a novel trajectory that is different from captured videos is a cornerstone for training autonomous vehicles. GEN3C can be applied.

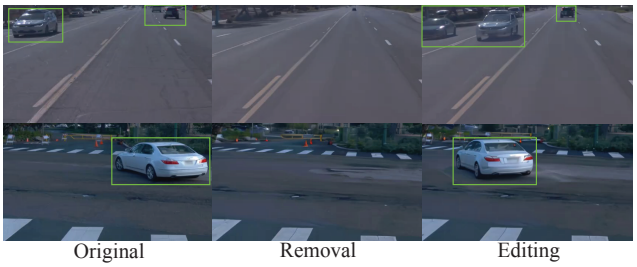


Figure 8. Qualitative results on 3D editing for driving scene. We remove and modify the trajectory of cars from the original scene.

Figure 9. Qualitative results on monocular dynamic NVS. Our model generates plausible novel camera trajectories for the given dynamic video. Note that, when zooming out from the original video, our model successfully reasons depth of field out.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓
GCD [54]	19.37	**0.67**	0.48	150.64
GEN3C	**19.41**	0.63	**0.29**	**98.58**

Table 4. Quantitative results on monocular dynamic NVS.

Evaluation and Baselines. We compare GEN3C to two representative scene reconstruction methods, Nerfacto [51] and 3DGS [25]. For a fair comparison, we filter out 18 static scenes from the validation set. For evaluation, we create the novel trajectories by horizontally shifting from the original trajectory of the frontal camera and varying the deviation. We report FID [18] for this evaluation since there is no ground truth for novel trajectories.

Results. As shown in Table 3, our method achieves significantly better FID scores than reconstruction methods on driving scenarios. This is because reconstruction methods struggle to recover the scene structures from the sparsely observed views in the driving scenario. Thus, the rendering quality degrades significantly when the rendering camera moves away from the original trajectory, as shown in Fig. 7.

3D Editing. Our explicit 3D cache naturally lends itself to 3D editing. As shown in Fig. 8, we can remove the 3D cars, modify the trajectory of the car, and generate a plausible re-simulation video of the driving scene using GEN3C.

5.5. Monocular Dynamic Novel View Synthesis

Given a monocular video of a dynamic scene, GEN3C is able to "rerender" that video along a new camera trajectory.

Noise Ratio	PSNR ↑	SSIM ↑	LPIPS↓
0%	24.08 / 21.56	0.86 / 0.83	0.12 / 0.15
3%	22.39 / 21.00	0.83 / 0.81	0.16 / 0.18
10%	20.85 / 19.64	0.79 / 0.76	0.19 / 0.22
30%	18.52 / 17.91	0.72 / 0.70	0.29 / 0.31

Table 5. Quantitative results on the robustness analysis on noisy depth estimation. The two values in each table cell represent the interpolation and extrapolation results, respectively.

	PSNR	SSIM	LPIPS
Explicit Fusion	21.81 / 19.87	0.79 / 0.75	0.21 / 0.28
Ours	**24.08 / 21.56**	**0.86 / 0.83**	**0.11 / 0.15**

Table 6. Ablation of different fusion strategies on RE10K dataset. The two values in each table cell represent the interpolation and extrapolation results, respectively.

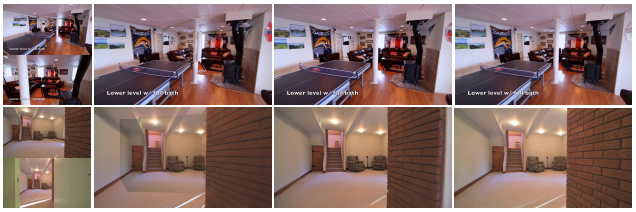


Figure 10. Qualitative results on ablating different fusion strategies. GEN3C can generate a realistic novel view with misaligned depth and different lighting in the input views, while the explicit fusion strategy fails.

Evaluation and Baselines. We evaluate on the 20 held-out test scenes of Kubric dataset released by GCD [54] and compare them to GCD. We use the publicly released checkpoint trained on Kubric datasets. Since GCD is only trained at a resolution of 256x384, we upsample its predictions to the same resolution as our method for a fair comparison.

Results. We provide quantitative results in Table 4, with qualitative results in the Supplement. Our method demonstrates superior performance in preserving object details and dynamics in input videos, and precisely aligns with the new camera motion from the users with a 3D cache.

Out-of-Domain Results. We further qualitatively evaluate GEN3C on dynamic videos generated by Sora [40] and MovieGen [35], and provide the results in Fig. 9. GEN3C generates photorealistic videos that preserve the 3D content and align with the new camera motion. We refer the readers to the supplement video for full results.

5.6. Ablation Study

We ablate our method in two ways: First, different strategies to fuse the point cloud as discussed in Sec. 4.3, and second, the robustness to noisy depth estimation. We follow the experiment settings from Sec. 5.3.

Different Fusion Strategies. We select two input views and predict the interpolation between these two views. Our fusion strategy is compared to the explicit fusion of the point

Figure 11. Qualitative comparison on using different base models: Stable Video Diffusion (SVD) [4] v.s. Cosmos [1]. When having a more powerful video generation model, GEN3C is able to generate more realistic output with less artifacts. Note that the slight misalignment between the two results is due to the models using different video resolutions.

Figure 12. Example of extreme NVS using Cosmos as the base model: the input view is the middle one, and our model is capable of rotating significantly to the left and right.

clouds from two views, in analogy to the method proposed in the concurrent work ReconX [30] and ViewCrafter [70]. Qualitative examples are provided in Fig. 10 with quantitative comparison in Table 6. Our method can smoothly transition between two disjoint views even if the depth estimates are misaligned and the lighting is different, while explicit fusion on the point cloud suffers from severe artifacts in misalignment regions.

Robustness to Noisy Depth Estimation. Since our model leverages an off-the-shelf depth estimator to create the 3D cache, we need to understand the impact of error in the depth estimation. Given a depth estimation, we add a noise sampled from the Gaussian distribution, $\mathcal{N}(0, s * (d_{0.95} - d_{0.05}))$. $d_{0.95}, d_{0.05}$ denote the 95 and 5 percentile of depth, respectively, and s is the noise ratio. We vary the noise ratio and evaluate the capability of producing novel views. As shown in Table 5, when having a small amount of noise, the performance drop is negligible, and our model still performs reasonably well when the noise ratio is large (30%).

5.7. Extending to Advanced Video Diffusion Model

We further replace the Stable Video Diffusion model [4] with Cosmos [1], a more advanced video diffusion model which has demonstrated superior performance in video generation. We follow the same fine-tuning protocol as above. Specifically, we chose `Cosmos1.0Diffusion-7BVideo2World`[1] as our base model and concatenate the noisy latent with the embedding of rendered frames, which are encoded by the Cosmos tokenizer. The model is fine-tuned on both the RE10K [75] and DL3DV [29] datasets for 10,000 steps with a batch size of 64.

We provide a qualitative comparison in Figure 11 with additional results on our webpage[2]. Results for extreme novel view synthesis are shown in Figure 12.

When leveraging a more powerful video diffusion model, GEN3C is able to generate videos with much higher quality, even under extreme camera viewpoint changes. This highlights a key strength of our method: its ability to leverage continuously evolving, pre-trained video models to achieve generalizability with minimal data required for fine-tuning.

6. Conclusion

In this paper, we introduced GEN3C, a consistent video generative model with precise camera control. We achieve this goal by constructing a 3D cache from seed image(s) or previously generated videos. We then render the cache into 2D videos from a user-provided camera trajectory to strongly condition our video generation, achieving more accurate camera control than previous methods. Our results are also state-of-the-art in sparse-view novel view synthesis, even in challenging settings such as driving scenes and monocular dynamic novel view synthesis.

Limitations. When generating videos with dynamic content, GEN3C relies on a pre-generated video to provide the motion of the objects. Generating such a video is a challenge on its own. A promising extension is to incorporate text conditioning to prompt for motion when training video generation models.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 9

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: a multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021. 2

[1] https://github.com/NVIDIA/Cosmos

[2] https://research.nvidia.com/labs/toronto-ai/GEN3C/

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2

[4] 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. 2, 3, 5, 9, 14

[5] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023. 3

[6] 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, 7, 14

[7] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVS-NeRF: fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, pages 14124–14133, 2021.

[8] 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, 7, 14

[9] Boyang Deng, Richard Tucker, Zhengqi Li, Leonidas Guibas, Noah Snavely, and Gordon Wetzstein. Streetscapes: Large-scale consistent street view generation using autoregressive video diffusion. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 3

[10] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: fewer views and faster training for free. In *CVPR*, pages 12882–12891, 2022. 2

[11] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4D-rotor Gaussian splatting: Towards efficient novel view synthesis for dynamic scenes. In *Proc. SIGGRAPH*, 2024. 2

[12] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T. Barron, and Ben Poole. CAT3D: create anything in 3D with multi-view diffusion models. *arXiv preprint arXiv:2405.10314*, 2024. 2

[13] 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. 3

[14] 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. 6, 16

[15] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. AnimateDiff: animate your personalized text-to-image diffusion models without specific tuning, 2023. 3

[16] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. CameraCtrl: enabling camera control for text-to-video generation. *CoRR*, abs/2404.02101, 2024. 2, 6, 7

[17] Roberto Henschel, Levon Khachatryan, Daniil Hayrapetyan, Hayk Poghosyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Streamingt2v: Consistent, dynamic, and extendable long video generation from text. *arXiv preprint arXiv:2403.14773*, 2024. 3

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 7, 8

[19] Chen Hou, Guoqiang Wei, Yan Zeng, and Zhibo Chen. Training-free camera control for video generation. *arXiv preprint arXiv:2406.10126*, 2024. 3

[20] Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. DepthCrafter: generating consistent long depth sequences for open-world videos. *arXiv preprint arXiv:2409.02095*, 2024. 4

[21] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. ACM, 2024. 2

[22] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: a large view synthesis model with minimal 3D inductive bias, 2024. 2

[23] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 3

[24] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 4

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2, 4, 7, 8

[26] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 6, 7

[27] Zhengfei Kuang, Shengqu Cai, Hao He, Yinghao Xu, Hongsheng Li, Leonidas Guibas, and Gordon Wetzstein. Collaborative video diffusion: Consistent multi-video generation with camera control. *arXiv preprint arXiv:2405.17414*, 2024. 3

[28] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H. Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8456–8465, 2023. 2

[29] 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. 6, 9, 14

[30] Fangfu Liu, Wenqiang Sun, Hanyang Wang, Yikai Wang, Haowen Sun, Junliang Ye, Jun Zhang, and Yueqi Duan. ReconX: reconstruct any scene from sparse views with video diffusion model. *arXiv preprint arXiv:2408.16767*, 2024. 2, 5, 9

[31] 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. 2, 3

[32] Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017. 14

[33] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2

[34] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *Proceedings of the European Conference on Computer Vision*, 2020. 3

[35] Meta. Meta Movie Gen: a cast of media foundation models. https://ai.meta.com/research/movie-gen/. Accessed: 2024-11-05. 2, 8

[36] 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. 2

[37] Norman Müller, Katja Schwarz, Barbara Rössle, Lorenzo Porzi, Samuel Rota Bulò, Matthias Nießner, and Peter Kontschieder. Multidiff: Consistent novel view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10258–10268, 2024. 2

[38] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 2

[39] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 2

[40] OpenAI. Sora: Creating video from text. https://openai.com/index/sora/. Accessed: 2024-11-05. 2, 8

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 5

[42] Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing A consistent long-term 3d scene video from A single image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 3553–3563. IEEE, 2022. 7

[43] Xuanchi Ren, Yifan Lu, Hanxue Liang, Zhangjie Wu, Huan Ling, Mike Chen, Sanja Fidler, Francis Williams, and Jiahui Huang. Scube: Instant large-scale scene reconstruction using voxsplats. *arXiv preprint arXiv:2410.20030*, 2024. 2

[44] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, pages 12892–12901, 2022. 2

[45] Runway. Runway. https://runwayml.com/. Accessed: 2024-11-05. 2

[46] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 3

[47] 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. 6, 14, 16

[48] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model, 2023. 2, 3

[49] Nagabhushan Somraj, Adithyan Karanayil, and Rajiv Soundararajan. SimpleNeRF: Regularizing sparse input neural radiance fields with simpler solutions. In *SIGGRAPH Asia*, 2023. 2

[50] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 6

[51] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 2, 7, 8

[52] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024. 2

[53] Zachary Teed and Jia Deng. DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras. *Advances in neural information processing systems*, 34: 16558–16569, 2021. 4, 6

[54] 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. 2, 3, 6, 8, 16

[55] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *ICCV*, pages 9065–9076, 2023. 2

[56] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2

[57] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: geometric 3D vision made easy. In *CVPR*, 2024. 2, 4

[58] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7

[59] Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. Adaptive shells for efficient neural radiance field rendering. *ACM Trans. Graph.*, 42(6), 2023. 2

[60] 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, 6, 14, 16

[61] Daniel Watson, Saurabh Saxena, Lala Li, Andrea Tagliasacchi, and David J. Fleet. Controlling space and time with diffusion models. *arXiv preprint arXiv:2407.07860*, 2024. 2, 3

[62] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, and Aleksander Holynski. ReconFusion: 3D reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21551–21561, 2024. 2

[63] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, and Yue Wang. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *arXiv preprint arXiv:2311.02077*, 2023. 2

[64] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *CVPR*, 2023. 2

[65] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 4

[66] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. *arXiv:2406.09414*, 2024. 4, 5, 6, 14

[67] 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. 3, 6, 14

[68] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

[69] Jason J Yu, Fereshteh Forghani, Konstantinos G Derpanis, and Marcus A Brubaker. Long-term photometric consistent novel view synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7094–7104, 2023. 7

[70] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. ViewCrafter: taming video diffusion models for high-fidelity novel view synthesis. *arXiv preprint arXiv:2409.02048*, 2024. 2, 5, 7, 9

[71] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. MonoSDF: exploring monocular geometric cues for neural implicit surface reconstruction. In *NeurIPS*, pages 25018–25032, 2022. 2

[72] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024. 2

[73] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7

[74] Kun Zhou, Wenbo Li, Yi Wang, Tao Hu, Nianjuan Jiang, Xiaoguang Han, and Jiangbo Lu. NeRFLix: high-quality neural view synthesis by learning a degradation-driven inter-viewpoint mixer. In *CVPR*, pages 12363–12374, 2023. 2

[75] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 6, 7, 9, 14

[76] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. FSGS: real-time few-shot view synthesis using Gaussian splatting. In *ECCV*, 2024. 2

Appendices

In the supplement, we provide additional details of our method in Sec. A and experiments in Sec. B. We provide more qualitative results in Sec. C.

A. Method Details

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

A.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 [66], 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:

$$s, t = \underset{s,t}{\operatorname{argmin}} \left\| \left(s \cdot \mathbf{d} + t - \mathbf{d}^{\text{tgt}}\right) \cdot M \right\|_2^2. \tag{4}$$

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

$$\mathbf{d}' = s \cdot \mathbf{d} + t. \tag{5}$$

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.

B. Experimental Details

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

B.1. Optimization Details

We optimize the neural network using AdamW optimizer [32] 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 [75] and DL3DV [29] 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.

B.2. Inference Details

We follow the practices from Stable Video Diffusion [4] and use classifier-free guidance during inference with the guidance weight being 3. We run 25 diffusion steps to generate the result.

B.3. Single-view to video generation

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

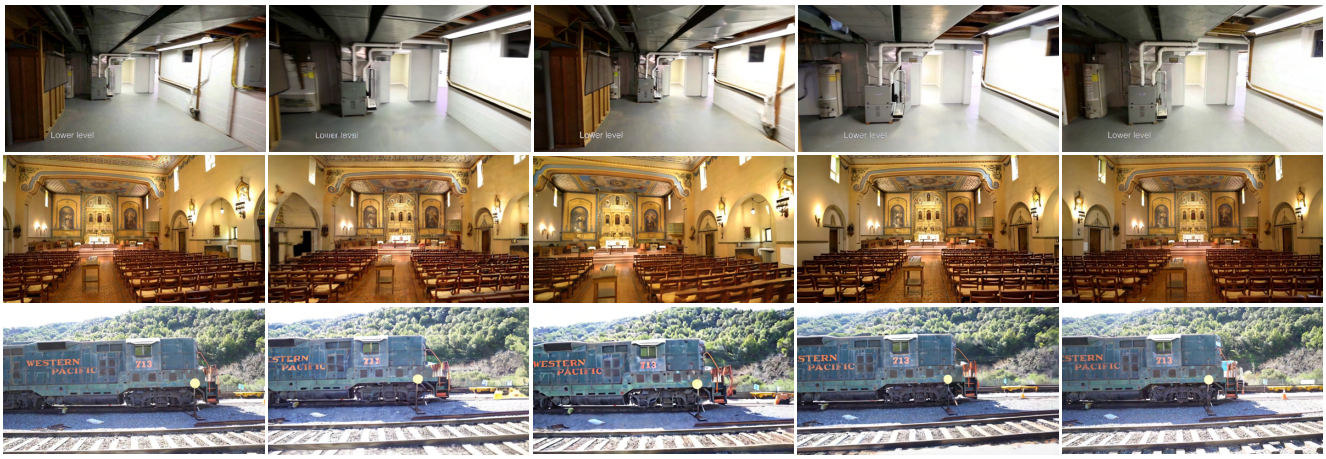
Baseline Details. For GenWarp [47][3] and MotionCtrl [60][4], we use the official checkpoint that is trained with Stable Video Diffusion [4] and evaluate on the same testing scenes as our method. Note that RE10k [75] is the training dataset for two methods. For NVS-Solver [67][5], we use the official codebase and run the evaluation using our testing data since the model is training-free.

B.4. Two-views NVS

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

Baseline Details. For PixelSplat [6], we take the official codebase and the released checkpoint[6] that is trained on RE10k [75] for a fair evaluation. For MVSplat [8], we also take the official codebase and the released checkpoint[7]. 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.

[3] https://github.com/sony/genwarp
[4] https://github.com/TencentARC/MotionCtrl
[5] https://github.com/ZHU-Zhiyu/NVS_Solver
[6] https://github.com/dcharatan/pixelsplat
[7] https://github.com/donydchen/mvsplat



Input GenWarp MotionCtrl Ours GT

Figure 13. Additional qualitative results for single-view novel view synthesis.

Input Ours

Figure 14. 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).

We run the two baselines on the testing data we prepared and report the results for both interpolation and extrapolation.

B.5. NVS for driving simulation

Baseline Details. For Nerfacto, we take the official codebase released by Nerfstudio[8], which is a state-of-the-art codebase for training Nerf. For 3DGS, we take the official codebase[9]. 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 cache is then used to generate the video at the novel camera trajectory.

B.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.

C. Additional Results

C.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. 16. 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.

[8] https://github.com/nerfstudio-project/nerfstudio

[9] https://github.com/graphdeco-inria/gaussian-splatting

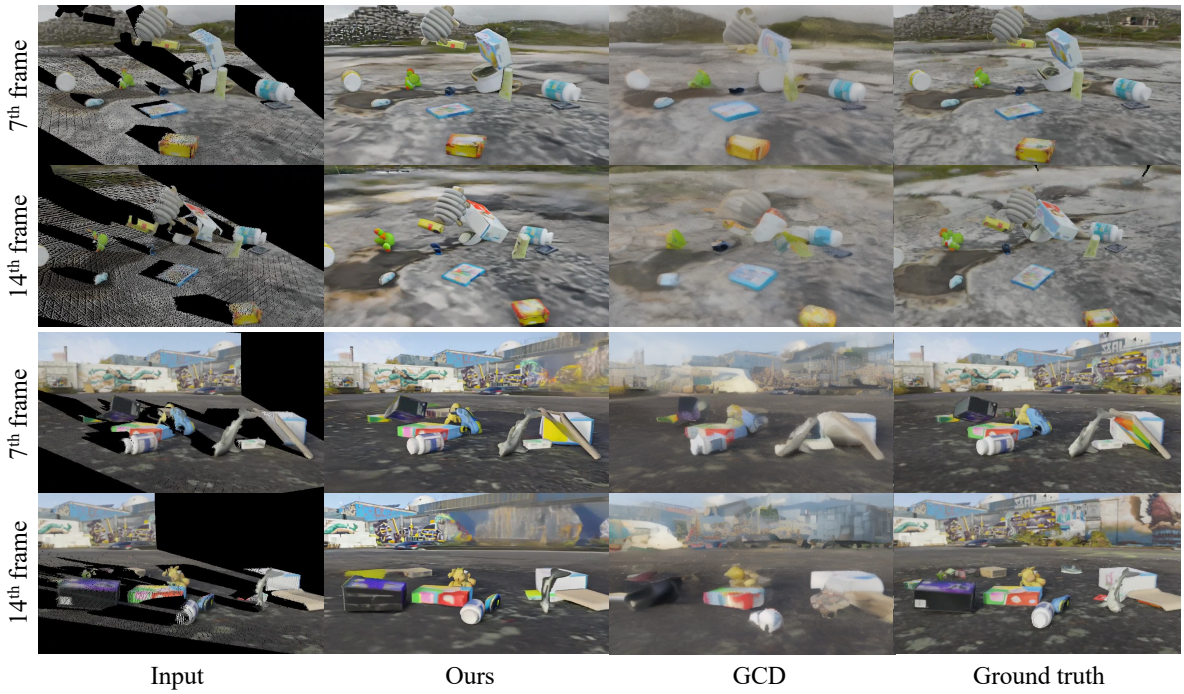


Figure 15. Qualitative results on Kubric4D [14]. Compared to GCD [54], our method demonstrates superior performance in preserving object details and dynamics in input videos.

Figure 16. 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.

C.2. Single-view to video generation

In addition to the comparison in the main paper, we provide the quantitative comparisons with GenWarp [47] and MotionCtrl [60] in Fig. 13. We also provide the rendered depth images and the model outputs in Fig. 14 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.

C.3. Monocular Dynamic Novel View Synthesis

We provide the qualitative comparison with GCD [54] on the Kubric dataset in Fig. 15. Our method generates sharper video details with more object details and dynamics compared to GCD [54].