4Real-Video-V2: Fused View-Time Attention and Feedforward Reconstruction for 4D Scene Generation

Chaoyang Wang[1*] **Ashkan Mirzaei** [1*] **Vidit Goel**[1] **Willi Menapace**[1] **Aliaksandr Siarohin**[1]
Avalon Vinella[1] **Michael Vasilkovsky**[1] **Ivan Skorokhodov**[1] **Vladislav Shakhrai**[1]
Sergey Korolev[1] **Sergey Tulyakov**[1] **Peter Wonka**[1,2]
[1]Snap Inc. [2]KAUST
Project page: https://snap-research.github.io/4Real-Video-V2/

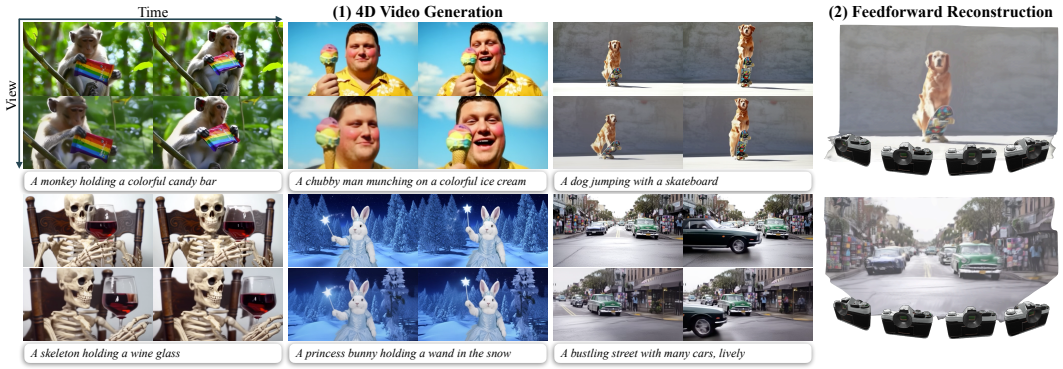

Figure 1: Our method enables the creation of 4D scenes from a text prompt by combining a diffusion model that directly generates synchronized multi-view videos with a feedforward reconstruction model that efficiently produces Gaussian-based representations.

Abstract

We propose the first framework capable of computing a 4D spatio-temporal grid of video frames and 3D Gaussian particles for each time step using a feed-forward architecture. Our architecture has two main components, a 4D video model and a 4D reconstruction model. In the first part, we analyze current 4D video diffusion architectures that perform spatial and temporal attention either sequentially or in parallel within a two-stream design. We highlight the limitations of existing approaches and introduce a novel fused architecture that performs spatial and temporal attention within a single layer. The key to our method is a sparse attention pattern, where tokens attend to others in the same frame, at the same timestamp, or from the same viewpoint. In the second part, we extend existing 3D reconstruction algorithms by introducing a Gaussian head, a camera token replacement algorithm, and additional dynamic layers and training. Overall, we establish a new state of the art for 4D generation, improving both visual quality and reconstruction capability.

1 Introduction

Immersive visual experiences are becoming increasingly popular in fields such as virtual reality and film production. This growing demand drives the need for technologies that enable the creation of 4D content, where users can interactively explore dynamic scenes. A key challenge in enabling such capabilities is the limited availability of high-quality 3D and 4D data. This scarcity presents a significant obstacle to training generative models that can directly produce 4D representations.

[*]Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Table 1: Comparison of recent 4D video generation methods. We use *Modified* to denote methods that rely on sophisticated adjustments to the diffusion process for generating 4D videos. A question mark (?) indicates methods that are theoretically extendable to 4D video generation, but such extensions were not explored in their original papers.

Category	Method	Model Output	Scene Type	4D Inference	DiT-based	Temporal Compress
4D-Aware Image/Video Gen	4DiM [1]	Images	Scene	?	✗	✗
	CAT4D [2]	Images	Scene	Modified	✗	✗
	GenXD [3]	Video	Scene & Object	Modified	✗	✗
	DimensionX [4]	Video	Scene	Modified	✓	✓
3D Point Cloud Cond. Video Gen	GEN3C [5]	Video	Scene	?	✓	✓
	TrajectoryCrafter [6]	Video	Scene	?	✓	✓
Video-Video Gen	Generative Camera Dolly [7]	Video	Scene	?	✗	✗
	ReCamMaster [8]	Video	Scene	?	✓	✓
4D (Sychronized Multi-View) Video Gen	CVD [9]	MV Video	Scene	Native (2 View)	✗	✗
	Human4DiT [10]	MV Video	Human	Native	✓	✗
	VividZoo [11]	MV Video	Object	Native	✗	✗
	4Diffusion [12]	MV Video	Object	Native	✗	✗
	SV4D [13]	MV Video	Object	Native	✗	✗
	SynCamMaster [14]	MV Video	Scene	Native	✓	✓
	4Real-Video [15]	MV Video	Scene	Native	✓	✗
	Ours	MV Video	Scene & Object	Native	✓	✓

In contrast, video generation has made rapid progress in recent years [16, 17, 18, 19, 20], driven by large-scale datasets and advances in diffusion models. Building on this progress, several recent works [15, 2, 13, 12] extend video generation to the 4D domain by producing what we refer to as 4D videos. These are synchronized multi-view video grids that can supervise reconstruction methods to recover explicit 4D representations such as dynamic NeRFs [21] or Gaussian splats [22]. We also adopt this two-stage approach because it leverages the strong priors of pretrained video models and offers a promising path toward generalizable and photorealistic 4D generation.

In the first stage, as categorized in Tab. 1, some prior methods attempt to enhance 2D video models with camera and motion control [2, 3, 4, 1], 3D point cloud condioning [6, 5], or reference-video conditioning [8, 7]. However, these models do not natively generate synchronized multi-view outputs, which often results in degraded quality and reduced efficiency.

Other approaches attempt to directly generate the complete multi-view video grid, where each row corresponds to a specific freeze-time step and each column to a fixed viewpoint. Most of these methods [13, 10, 11, 12] are designed for object-centric data and do not generalize to complex scenes. Among the few general-purpose models, SynCamMaster [14] is trained to generate a sparse set of diverse viewpoints, but often suffers from poor consistency across views. 4Real-Video [15] achieves stronger multi-view alignment by training with dense viewpoints, but is only tested on a smaller architecture that lacks features of modern video models, such as temporal compression and higher resolution.

When scaling to modern architectures, it is important to consider their large number of parameters (e.g., we use an 11B-parameter base model). Given the limited availability of 4D video training data and the significantly increased number of tokens introduced by handling multiple viewpoints, the design of a 4D video model becomes critically important. The two main current architectures are the sequential architecture interleaving spatial and temporal attention [11, 14], and the parallel architecture that computes spatial and temporal attention in parallel and merges the results [15]. By experimenting with multiple architecture variations, we believe that the most important criterion for developing an architecture is to minimize the number of new parameters that need to be trained from scratch and to minimize fine-tuning, so that each layer is used similarly to its pretraining. Building on this analysis, our key contribution is a parameter-efficient design that introduces no additional parameters to the base model. Specifically, we fuse cross-view and cross-time attention into a single self-attention mechanism. In contrast to previous approaches that apply these attentions separately and introduce new attention [14, 10, 12, 11] or synchronization modules [15], our unified formulation enables us to leverage highly optimized sparse attention implementations. This results in minimal computational overhead and enables effective fine-tuning of large pretrained video models.

In the second stage, traditional reconstruction methods often rely on iterative optimization to recover 4D representations from multi-view video inputs. Although accurate, these methods tend to be slow, sensitive to camera estimation errors, and difficult to scale to dynamic scenes of longer duration. To address this, we extend a state-of-the-art feedforward 3D reconstruction [23] to directly predict both camera parameters and time-varying Gaussian splats from synchronized multi-view video frames. This approach greatly improves efficiency while preserving visual quality.

In summary, our contributions are: 1) A novel two-stage 4D generation framework that produces a grid of images and converts them into Gaussian ellipsoids. 2) A fused view and time attention mechanism that enables parameter-efficient 4D video generation. 3) A feedforward model that jointly recovers camera parameters and Gaussian particles from multi-view videos.

2 Related Work

Optimization-based 4D generation. Score Distillation Sampling (SDS)[24, 25, 26, 27, 28, 29] is a common method for creating 3D scenes. It uses gradients from pre-trained models like text-to-image [30, 31] and text-to-multi-view [32, 33] models. Recent 4D methods [34, 35, 36, 37, 38, 39, 40, 41] extend this by using text-to-video models [42, 43, 44] to add motion. These methods usually take hours to run because they rely on slow optimization. Most of them also use 3D priors from diffusion models [32, 33] trained on synthetic object-centric datasets like Objaverse [45], which can make the results look unrealistic and limited to single objects.

Camera-Aware video generation. Text-to-video models [46, 16, 19, 20, 47] have made significant progress in generating realistic videos. To provide users with more control, some methods incorporate camera motion by leveraging camera pose data [48, 49, 50, 1], while others fine-tune models using videos annotated with camera labels [3, 51, 4]. CVD [9] and ReCamMaster [8] further enable modifying camera motion of on existing footage. Another line of work introduces a 3D cache, such as a point cloud, to store scene geometry. These representations are then projected into novel views and completed using diffusion models [6, 5, 52]. These camera-aware techniques enable impressive 3D visual effects, such as dolly shots and bullet-time sequences. However, these methods do not natively generate a complete set of frames across both time and viewpoints. Extending them to 4D video requires substantial modifications to the diffusion process [2, 4], often leading to artifacts and quality degradation.

4D video generation. We define 4D video (or synchronized multi-view video) as a grid of video frames organized along both temporal and viewpoint dimensions. Several methods [13, 11, 12, 10] are trained on 4D datasets derived from Objaverse [45] or human motion capture sequences. While these models aim to generalize beyond single-object scenes, they are still limited by the lack of diverse and scalable 4D training data. CVD [9] addresses this limitation by fine-tuning models to generate synchronized video pairs using pseudo-paired samples from real-world datasets [53, 54]. SynCamMaster [14] and 4Real-Video [15] further extend this direction by generating synchronized multi-view videos using a combination of synthetic 4D and real 2D datasets. SynCamMaster is trained on sparsely sampled viewpoints, allowing for good view control but showing inconsistencies across views. 4Real-Video, on the other hand, uses densely sampled, continuous camera trajectories to improve view consistency. However, it is built on a relatively small pixel-based video backbone, which limits visual quality and lacks temporal compression, making it inefficient for longer videos. Our work improves upon 4Real-Video by introducing a more efficient architecture that scales effectively with large video generation models.

Feed-forward reconstruction. Recent advances in 3D reconstruction increasingly use data-driven priors to speed up the process [55]. Some methods use priors for guidance or initialization to enable faster optimization [56, 57, 58], but still rely on few-shot optimization to refine results [59, 60]. To avoid optimization, newer work explores fully feedforward models that infer 3D scenes from 2D inputs. These models often focus on static scenes and represent geometry using triplanes [61], 3D Gaussians [62, 63, 64, 65], sparse voxel grids [66], or learned tokens [67, 68]. They usually need ground-truth camera calibration or rely on traditional methods to estimate camera parameters, which can be slow and unreliable for generated assets. This has driven interest in pose-free, feedforward reconstruction for static scenes [69, 70, 71, 72]. While these models work well for static scenes, handling dynamic scenes is still a challenge. Current dynamic scene methods often assume dense, temporally consistent video depth maps [73], which are hard to obtain. Others lack rendering support [74, 75, 76, 23], or only work with object-centric data [77]. Some also assume known camera poses and monocular video [78]. Another challenge is that even when RGB loss is used for realistic outputs, many methods produce poor geometry, limiting novel view synthesis to small camera movements [65, 78].

3 Method

Our 4D generation pipeline comprises two main stages. First, we introduce a novel 4D video diffusion model that generates synchronized multi-view videos of dynamic scenes across time and viewpoints

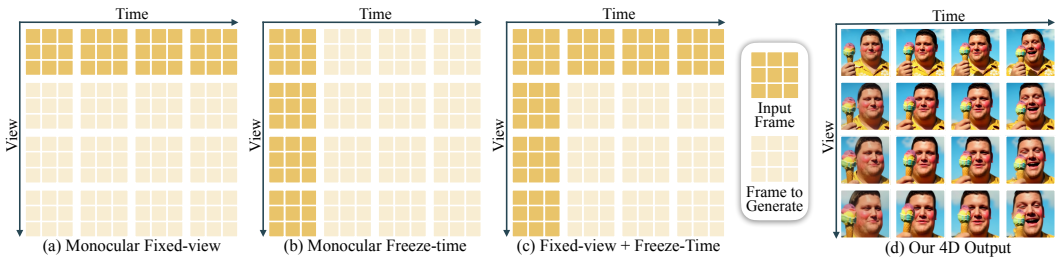

Figure 2: Our 4D video model supports input types including: (a) a fixed-view video, (b) a freeze-time video showing multiple angles of a scene at a single timestep, and (c) a combination of both. Each input can be generated from a text prompt using standard video models.

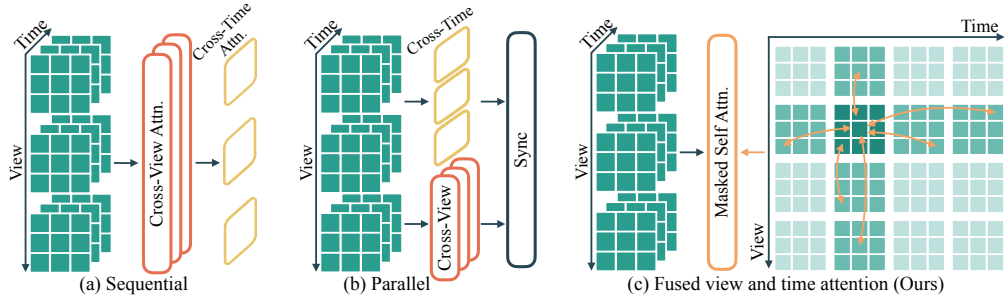

Figure 3: We analyze three architectures for 4D video generation: (a) sequential cross-view and cross-time attention, (b) parallel cross-view and cross-time attention with an extra synchronization layer, and (c) our proposed architecture using fused view-time attention with masked self-attention.

(Sec. 3.1). Next, we apply a purpose-built feedforward reconstruction network to lift Gaussian ellipsoids from the generated frames (Sec. 3.2). We detail each component below.

3.1 Synchronized Multi-view Video Diffusion

We aim to generate a structured grid of video frames $\{I_{v,t}\}$, where all frames in a row share a viewpoint v, and all frames in a column share a timestep t. In other words, each row is a fixed-view video, and each column is a freeze-time video.

Preliminary I: DIT-based Diffusion Transformer. The Diffusion Transformer (DiT) [79] architecture has been widely adopted in modern video diffusion models [16, 17, 80, 19, 18] for denoising latent video tokens. The model takes as input a set of latent tokens $\{\mathbf{x}_{t,x,y} \in \mathbb{R}^d\}$, where t, x, and y index the temporal and spatial dimensions, and d is the latent dimension. These tokens represent compressed versions of high-resolution videos, typically downsampled by a factor of $8\times$ spatially and $4\times$ or $8\times$ temporally. The tokens are first embedded via a patch embedding layer and then processed through a series of DiT blocks. Each block contains a 3D self-attention layer that jointly attends the features across both spatial and temporal dimensions, followed by a cross-attention layer that conditions the features on input context, such as text embeddings.

Preliminary II: Prior architectures for synchronized multi-view video diffusion. Standard video diffusion models generate a single sequence of frames with entangled view changes and scene motion. To extend pretrained video diffusion models for generating synchronized multi-view videos, prior works introduce additional cross-view self-attention layers to enforce consistency across views. As illustrated in Fig. 3, these methods can be categorized based on how the cross-view attention is incorporated into the architecture.

Sequential architecture. These methods [11, 14, 10, 12, 13] *sequentially* interleave cross-view and cross-time self-attention layers. The cross-view layers are typically initialized from multi-view image models trained to synthesize static scenes from different viewpoints, while the cross-time layers are initialized from pretrained video models. These attention layers are either jointly trained [10, 13], or fine-tuned selectively by freezing one type while updating the other [14, 12].

Parallel architecture. An alternative strategy applies cross-view and cross-time self-attention in parallel, rather than interleaving them sequentially [15, 81]. In this setup, each attention branch

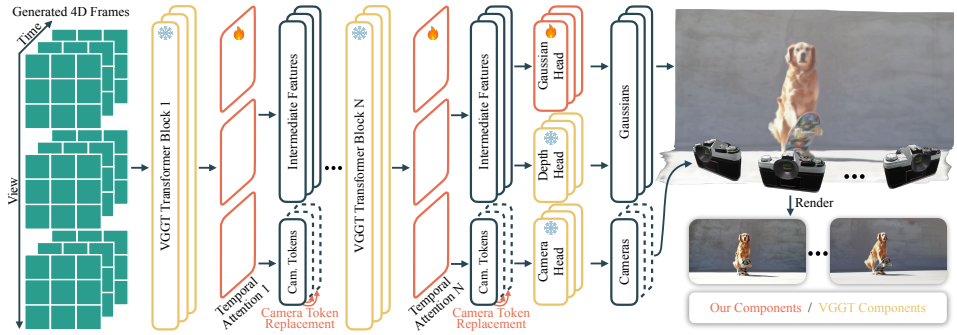

Figure 4: Overview of our feedforward reconstruction model. Built on top of VGGT [23], it incorporates temporal attention layers, camera token replacement to ensure consistent cameras over time, and a Gaussian head that predicts Gaussian parameters.

processes the video tokens independently—cross-view attention enforces spatial consistency across viewpoints, while cross-time attention captures temporal dynamics. The outputs from both branches are then fused through a synchronization module designed to align and integrate the two branches. This decoupled design has two key advantages: (1) it avoids interference between the attention branches, which are originally trained to operate in isolation, and (2) it enables reusing frozen pretrained models for both branches, reducing training cost and preserving their generalization capability. Only the lightweight synchronization module needs to be trained [15].

Fused view and time attention architecture. Parallel architectures introduce additional parameters through the synchronization module. To mitigate overfitting on limited 4D data, these modules are deliberately kept lightweight, typically implemented as a single linear layer [15] or a weighted averaging operation [81]. In contrast, we propose a new design that requires no additional parameters beyond a pretrained video model, and as a result, it naturally maintains strong generalization without relying on manually crafted bottleneck layers.

Specifically, we propose to fuse cross-view and cross-time attention into a single self-attention operation. For each latent token $\mathbf{x}_{v,t,x,y}$ representing view v, time t, and spatial location (x, y), features are computed by attending to all other tokens sharing either the same view or timestamp. This is implemented using a masked self-attention layer that enforces the desired attention pattern:

$$\text{SoftMax}(\mathbf{M} \odot \frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{d}})\mathbf{V}, \quad \mathbf{M}(\text{Idx}(v_q, t_q, x_q, y_q), \text{Idx}(v_k, t_k, x_k, y_k)) = \begin{cases} 1, & \text{if } v_q = v_k, \text{ or } t_q = t_k \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V} \in \mathbb{R}^{N \times d}$ are the queries, keys and values of all tokens, $N = VTHW$ is the total number of tokens, $\odot$ denotes element-wise multiplication, and $\mathbf{M} \in \mathbb{R}^{N \times N}$ is a binary mask. The function $\text{Idx}()$ maps the view, temporal and spatial indices of a token to its corresponding flattened index.

The masked self-attention is efficiently implemented using FlexAttention [82], which exploits the sparsity of the attention mask to reduce memory and computation. With a high sparsity ratio of $1 - \frac{T+V}{TV}$, the approach scales efficiently to a large number of views and timestamps.

Positional embedding. For standard 2D video generation, each token is associated with a 3D positional embedding (t, x, y). In contrast, multi-view videos introduce an additional view dimension, resulting in 4D indices (v, t, x, y). Directly using 4D positional embeddings would introduce significant discrepancies with the pretrained video model. To address this, we map the 4D coordinates to 3D by collapsing the view and time dimensions: $(v, t, x, y) \rightarrow (v * T_{\text{max}} + t, x, y)$, where T_{max} is the maximum temporal length supported by the model. This is equivalent to flattening the multi-view video into a single long sequence. A similar idea was independently adopted by ReCamMaster [8] for the specific case of synchronizing two videos. We also explored alternative transformations (see Supplementary), but found the above approach to be the most effective.

Temporally compressed latents. Each latent token compresses patches from 4 consecutive frames of the same view points. We choose to compress along the temporal dimension rather than the view dimension, as densely sampled viewpoints offer limited benefit to reconstruction quality in the second stage. In contrast, temporal compression significantly improves generation throughput.

Table 2: Quantitative comparison of our architecture (fused view-time attention) with baselines.

Method	Objaverse PSNR↑	Objaverse SSIM↑	Objaverse LPIPS↓	NVIDIA Dynamic Dataset [85] PSNR↑	NVIDIA Dynamic Dataset [85] SSIM↑	NVIDIA Dynamic Dataset [85] LPIPS↓
SV4D [13]	19.65	0.883	0.137	-	-	-
Sequential Arch.	5.935	0.444	0.583	22.74	0.743	0.147
Parallel Arch.	21.40	0.892	0.124	22.92	0.742	0.148
Fused View & Time Attn.	**22.49**	**0.909**	**0.113**	**23.15**	**0.752**	**0.142**

Grounded generation with reference videos. Inspired by prior works [15, 13], we enable the model to condition on two types of reference videos: a *fixed-view* video, which specifies the content and object motion of the dynamic scene from a single viewpoint, and a *freeze-time* video, which captures the scene from multiple views at a single timestamp (see illustration in Fig. 2). The model then generates all other frames corresponding to unseen view-time combinations. The conditioning is implemented by adding the patch embeddings of the reference frames to the noised latent tokens before feeding them into the denoising transformer. Unlike prior works that rely on explicit camera pose embeddings, our model learns to infer viewpoints directly from the input reference videos.

This design choice improves usability for 3D scene animation tasks, as it avoids the need for explicit pose estimation and scale alignment, which are often technically challenging for non-expert users.

Training. The model is trained using the same rectified flow objective [83] as the base video model, following a progressive schedule that gradually increases the temporal duration. Only the self-attention layers are fine-tuned, while the rest of the model remains frozen. Training is efficient, and we report results after 4k iterations with a batch size of 96. Additional model details are provided in the supplementary. The training data includes: 1) *Synthetic multi-view videos*, rendered using animated 3D assets [45] and physics-based simulations [84]. 2) *2D transformed videos*, created by applying random 2D homographic transformations to each video frame to simulate synchronized multi-view captures. This data is a key augmentation for the limited real 4D data. 3) *3D videos*, depicting static scenes or objects recorded along continuous camera trajectories. These are temporally duplicated to simulate multi-view sequences without dynamic motion.

3.2 Feedforward Reconstruction

Our 4D video generator synthesizes visual content, which is then passed to our Gaussian-based [22] feedforward reconstruction model. This model operates on RGB frames only, since camera parameters are not provided. To recover geometry and camera parameters, we use a pretrained VGGT model [23]. VGGT is a transformer-based neural network for 3D scene understanding that processes multi-view images using DINO-encoded tokens and learnable camera tokens. It predicts camera parameters and dense 3D outputs such as depth maps and point clouds, all in a canonical frame aligned to the first camera. We unproject depth maps into 3D point clouds, which serve as Gaussian centroids. A DPT-based head, called the Gaussian head, is trained to estimate the remaining Gaussian parameters: opacities, scales, and rotations. Colors are derived from rays in RGB space and refined with residuals predicted by the Gaussian head. An overview of this process is shown in Figure 4.

Camera Token Replacement. To extend the model to dynamic scenes, applying it independently to each frame causes inconsistent camera predictions. To enforce temporal consistency, we replace the camera tokens of all views at each timestep with those from the first timestep, after the VGGT transformer blocks. This ensures that all frames share the same predicted camera parameters and improves consistency over time.

Gaussian Head. The Gaussian head predicts opacities, rotations, and scales from refined image tokens. Centroids are derived from unprojected depth maps, with a predicted 3-dimensional pose-refinement which is added to the unprojected depths (following Splatt3r [69]). We train this module using a reconstruction loss composed of MSE and LPIPS: $\mathcal{L}_{\text{recon}} = \mathcal{L}_{\text{MSE}} + \lambda_{\text{LPIPS}}\mathcal{L}_{\text{LPIPS}}$, where the perceptual loss encourages photometric fidelity [86].

Temporal Attention. The VGGT backbone uses Alternating-Attention layers to mix global and frame-wise information. After each frame and global attention layer, we add a temporal attention layer to connect tokens across timesteps. This helps the model share information across time in dynamic scenes. The temporal attention layer is zero-initialized so the model's initial predictions match the original VGGT outputs.

Training. The first training stage uses both synthetic and real-world static datasets. We include RealEstate10K [54], DL3DV [87], MVImageNet [88], Kubric [84] (only single-timestep samples),

Table 3: Quantitative comparison with baselines on the generated video dataset.

Method	Cross-View	Cross-Time (VBench [90])				
	Met3R↓ [91]	Flickering↑	Motion↑	Subject↑	Background ↑	Image↑
TrajectoryCrafter [6]	0.324	97.1	98.5	95.3	96.8	**67.6**
SynCamMaster [14]	0.530	**99.3**	**99.5**	97.2	96.6	65.7
ReCamMaster-V1 [8]	0.530	98.6	99.4	96.6	96.1	66.0
ReCamMaster-V2 [8]	0.194	94.7	91.2	90.7	93.6	65.4
4Real-Video [15]	0.192	98.7	99.2	94.4	96.5	64.4
w/o 2D Trans. Videos	0.196	**99.3**	**99.5**	**98.0**	**98.7**	63.9
Full Method	**0.173**	99.1	**99.5**	97.7	98.4	66.2

and ACID [89]. Each iteration samples scenes and views, predicts Gaussian parameters, renders the scene, and computes the loss. The VGGT backbone stays frozen to reduce memory use and allow larger batch sizes. This setup simplifies training by letting VGGT handle geometry and color, while the Gaussian head learns only residual parameters. We then train on dynamic Kubric to tune the temporal attention layers, while continuing to finetune the Gaussian head. We also reuse static datasets at this stage by copying multi-view samples across timesteps to create static 4D datasets, which helps prevent forgetting. In both stages, we use 4 source views. For dynamic training, we sample 4 views at 4 timesteps each. The rest of the training hyperparameters follow BTimer [78]. More details are provided in the supplementary.

4 Experiments

4.1 Synchronized Multi-View Video Generation Evaluation

Evaluation datasets. We evaluate the 4D video generation capability across a combination of datasets: 1) *Generated videos.* We run Veo 2 [20] to create 30 fixed-view videos, each prompted by a unique caption. To enforce a static viewpoint, we append the phrase *"static shot. The camera is completely static and doesn't move at all."* to each caption. We then extract the first frame of each generated video and duplicate it to create a static video. This static video is passed to ReCamMaster [8] to produce a freeze-time video (reference freeze-time) of the scene. This process establishes the first column and first row of our 4D grid, which we keep consistent across all baselines that utilize it. 2) *Objaverse.* Following SV4D, we collect 19 animated 3D assets from Objaverse [45] that are not included in the training set. 3) *Nvidia Dynamic Dataset.* This dataset [85] contains 9 dynamic scenes captured by 12 synchronized cameras, offering real-world multi-view data.

Baselines. We compare against state-of-the-art video generation methods that either natively support synchronized multi-view generation or can be adapted to do so: 1) *TrajectoryCrafter* [6] is a representative baseline for point cloud-conditioned methods. As a 2-view model, it generates fixed target views conditioned on the reference fixed-view video. We use it to produce 8 distinct views per scene. 2) *ReCamMaster* [8] generates a video with a modified camera trajectory, conditioned on a reference video that shares the first frame with the output. Since it is not directly suitable for multi-view generation, we adapt it in two variants: *ReCamMaster-V1*: We construct a pseudo-static reference video by repeating the first frame for the first half, followed by the original freeze-view video. The target camera trajectory moves during the first half and remains static in the second. We retain only the second half of the output, yielding an approximately fixed-view rendering from a new viewpoint. *ReCamMaster-V2*: We generate independent freeze-time videos for each timestep by conditioning on static reference videos, created by repeating the corresponding frames of the input fixed-view video. 3) *SynCamMaster* [14] & *4Real-Video* [15] are both multi-view models. Since the released SynCamMaster code does not support frame conditioning, we evaluate it using text conditioning. 4) *SV4D* [13] is a multi-view generation model trained on object-centric data.

Evaluation protocal. For all datasets, we prepared fixed-view videos as reference inputs and freeze-time videos to condition the view points for generation. For methods that do not support frame-level conditioning, we instead provide camera poses. On datasets with ground truth frames (Objaverse and NVIDIA Dynamic), we evaluate reconstruction quality using standard metrics: PSNR, SSIM, and LPIPS. For the generated video dataset, we subsample outputs into fixed-view and freeze-time videos. We assess multi-view consistency using Met3R [91], and evaluate visual quality of the fixed-view videos using the widely adopted VBench [90] metrics.

Comparing model architectures. We compare three model architecture variants (see Fig.3): *sequential*, *parallel*, and our proposed fused view-time attention. All models are trained under identical settings for 4,000 iterations with a batch size of 96.

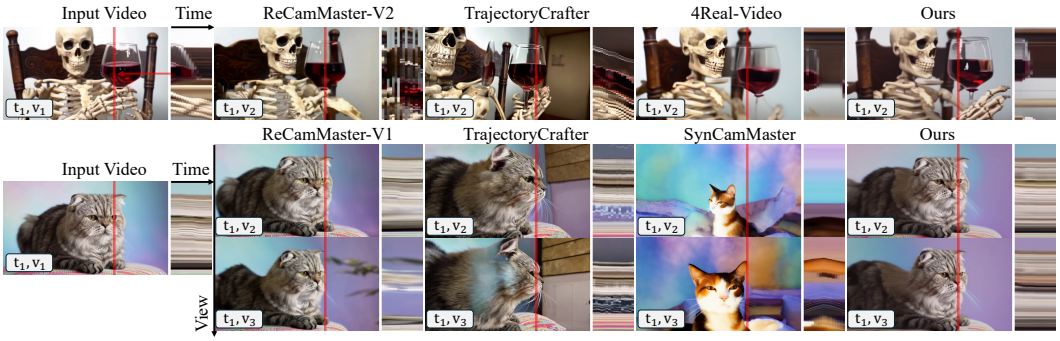

Figure 5: Visual comparison of 4D video generation methods. Each image includes a temporal slice (right) along the red line to reveal temporal flickering. ReCamMaster-V2 shows strong flickering; ReCamMaster-V1 has inconsistent backgrounds across views. TrajectoryCrafter exhibits artifacts from noisy point clouds. 4Real-Video misses thin structures, and SynCamMaster produces inconsistent synthetic-style results. Our method achieves the best visual quality and consistency.

Table 4: Computation efficiency comparison of multi-view video generation architectures, profiled on an A100 GPU (float32) for a single forward pass with 8 views and 61 frames at 288×512 resolution.

	Sequential (Cross-View Attn.)	Sequential (Interleave Existing Layers)	Parallel	Fused V & T Attn. (wo Block Mask)	Fused V & T Attn. (with Block Mask)
Parameters	17B	11B	11B	11B	11B
Peak Memory (GB)	OOM	54.9	55.9	54.9	54.9
Runtime (s)	OOM	9.3	13.9	20.5	11.7

Generation quality. We evaluate their performance quantitatively on the Objaverse and NVIDIA Dynamic datasets (see Tab.2). Our proposed fused view-time attention consistently outperforms the other variants. The parallel architecture shows a noticeable drop in performance, suggesting that the introduction of bottleneck (synchronization) layers may limit model capacity. The sequential architecture learns significantly slower and fails to generate proper white backgrounds for Objaverse scenes (See Fig.S1 for visualizations). We have identified the issue as an instability that occurs when the sequential model is trained on our mixed dataset. Its loss curve fluctuates more and converges to a higher value, reflecting less reliable/stable learning. Although the model performs reasonably well on real scenes (NVIDIA Dynamic dataset), it often misidentifies the white background which appears in about one-third of the training sequences, producing visible artifacts in the results. These background artifacts are simply the most obvious out of multiple failures; a closer inspection also reveals subtler issues such as blurred edges and temporal flickering. The sequential design exhibits an effective capacity that is lower than that of the fused model, making it harder to reconcile the differing background statistics.

Computation efficiency. We measured the runtime and peak memory consumption across different architectural designs. Profiling was performed on an A100 GPU using float32 precision, evaluating a single forward pass with 8 views and 61 frames at a resolution of 288×512 (corresponding to 8×16×36×64 tokens after tokenization and patchification). The results reveal the following: (1) a sequential architecture that introduces cross-view attention layers (as in SynCamMaster [14]) increases the total parameter count by approximately 50%, leading to out-of-memory errors given the already substantial 11B-parameter base video model; (2) a variant sequential model that reuses existing DiT blocks but interleaves cross-view and cross-time attention achieves faster runtime at the cost of degraded quality (see Tab. 2); (3) a parallel architecture (similar to 4Real-Video [15]) incurs higher peak memory usage and longer runtime; and (4) the proposed fused View-Time attention, when applied without block masking, runs significantly slower compared to its block-masked counterpart.

Training data sensitivity. We analyzed the sensitivity of multi-view video generation to different training datasets. Tab. 5 summarizes an ablation study across various training set combinations, evaluated on the object-centric Objaverse dataset and the dynamic-scene Nvidia dataset. All models were trained under identical conditions (batch size of 128, 2k iterations). Our findings are as follows: *Data efficiency:* Our method demonstrates strong data efficiency. For example, a model trained solely on Objaverse generalizes reasonably well to the out-of-distribution Nvidia dataset. *Importance of 2D-transformed video datasets:* These datasets play a crucial role in dynamic scene generation,

Table 5: Analysis of training data sensitivity in multi-view video generation.

Train Datasets			Objaverse			NVidia Dynamic Dataset		
Objaverse	Kubric	2D Trans. Vid.	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✓			**23.30**	**0.923**	**0.071**	22.77	0.729	0.130
	✓		17.99	0.858	0.172	21.95	0.706	0.165
		✓	19.52	0.881	0.125	23.02	0.736	0.130
✓	✓		22.38	0.914	0.086	22.74	0.727	0.136
✓	✓	✓	22.09	0.911	0.084	**23.38**	**0.742**	**0.124**

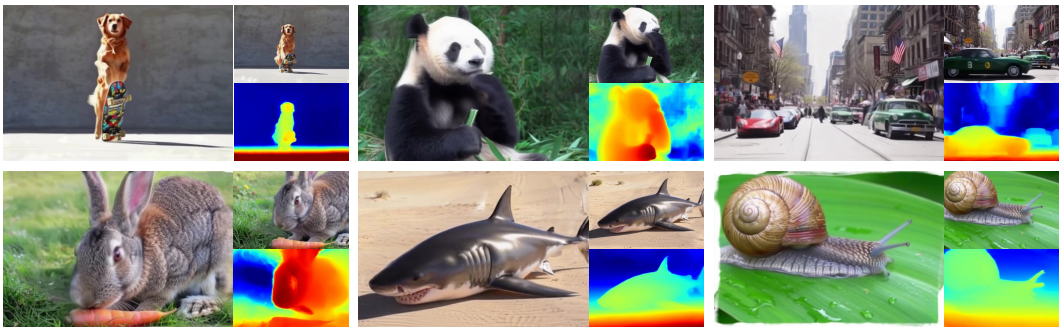

Figure 6: Color and depth renderings of Gaussians produced by inputting our generated 4D video grids into the reconstruction model.

producing the second-best results on Nvidia when used in isolation. Removing this data source leads to a noticeable degradation in video generation quality, as shown in Tab. 3. *Combined datasets:* Training with all datasets combined yields consistently strong performance across both scenarios.

Comparing baselines. On Objaverse, our method produces noticeably higher-quality results compared to SV4D, as shown in Tab. 2 (see the supplement for visuals). SV4D often generates blurry frames. We attribute this to limitations of its base model and exclusive training on synthetic data.

On the generated dataset, our method consistently outperforms all baselines in both video quality and multi-view consistency (see Tab.3 and Fig.5). The publicly released SynCamMaster model shows noticeable inconsistencies across views and exhibits a bias toward synthetic-style outputs. ReCamMaster-V1 struggles to maintain a static camera trajectory, and because each fixed-view video is generated independently, it lacks multi-view consistency. ReCamMaster-V2 achieves better multi-view alignment but suffers from temporal flickering. TrajectoryCrafter produces consistent outputs overall, but artifacts often emerge due to outliers in the conditioned point clouds, reducing visual fidelity. Lastly, 4Real-Video is constrained by its low-resolution, pixel-based model, resulting in degraded visual quality and frequent failure to render fine details, such as the fingers of the skeleton.

4.2 Feedforward Reconstruction Evaluation

We evaluate our feedforward reconstruction model independently, comparing it to state-of-the-art methods for static and dynamic scenes. See the supplement for details on the baselines, datasets, and metrics.

NVS for static scenes. Table 6 shows quantitative results comparing our method with the baselines (See the supplement for visualizations). GSLRM and BTimer lack scene-scale standardization, so their performance can vary if camera parameters are manually scaled for each test scene (using a tedious grid search). To ensure fairness, we report results for GSLRM/BTimer with and without per-scene manual scale tuning (see the “Manual Scale" column). Even with tuning, these methods do not match our performance. Our method does not use manual tuning, nor does it rely on input/output camera parameters. Instead, it predicts all camera parameters. This adds difficulty, as it can affect both Gaussian predictions and camera accuracy. We use VGGT [23] to predict the output camera parameters by taking the first input frame and the target frames. These predictions are made in the coordinate system of the first input, which is also the frame of reference for our Gaussians. Despite these challenges, our method still outperforms the baselines, even when they are manually tuned.

NVS for dynamic scenes. We compare our model to baselines on novel view synthesis for dynamic scenes using the Neural3DVideo [94] dataset, on which none of the baselines were trained. Figure 7 shows a visual comparison. GSLRM is designed for static scenes and is applied here by processing

Table 6: Quantitative comparison with baselines on static scene novel view synthesis.

Method	# Inputs	Input Cams.	Manual Scale	Tanks & Temples [92] PSNR↑	LPIPS↓	SSIM↑	LLFF [93] PSNR↑	LPIPS↓	SSIM↑
GSLRM [65]	4	Yes	Yes	15.78	0.3896	0.3385	14.42	0.4465	0.2980
GSLRM [65]	16	Yes	Yes	16.21	0.4236	0.3528	14.85	0.4919	0.3222
BTimer [78] (Static)	4	Yes	Yes	20.62	0.1498	0.5762	16.40	0.2789	0.3669
BTimer [78] (Static)	16	Yes	Yes	20.45	0.1633	0.5771	16.60	0.3225	0.3971
GSLRM [65]	4	Yes	No	12.41	0.5933	0.3038	10.69	0.6182	0.2785
GSLRM [65]	16	Yes	No	12.40	0.6138	0.3244	12.15	0.6323	0.2946
BTimer [78] (Static)	4	Yes	No	16.48	0.2781	0.3933	14.24	0.3958	0.2830
BTimer [78] (Static)	16	Yes	No	17.13	0.2883	0.4477	14.63	0.4231	0.3142
Splatt3r [69]	2	No	No	12.50	0.4547	0.3363	12.64	0.4599	0.3055
Ours	4	No	No	18.52	0.1699	0.5178	15.12	0.2778	0.3024
Ours	16	No	No	**20.85**	**0.1464**	**0.6057**	**18.95**	**0.1919**	**0.4573**

Table 7: Comparison with baselines for dynamic NVS on the Neural3DVideo [94] dataset.

Method	Input Cams.	Manual Scale	PSNR↑	LPIPS↓	SSIM↑
GSLRM [65]	Yes	Yes	20.54	0.1934	0.6346
BTimer [78] (monocular)	Yes	Yes	9.65	0.4310	0.3907
BTimer [78] (multi-view)	Yes	Yes	21.55	0.1213	**0.6412**
GSLRM [65]	Yes	No	12.31	0.5866	0.3553
BTimer [78] (monocular)	Yes	No	9.40	0.5898	0.3006
BTimer [78] (multi-view)	Yes	No	19.56	0.1693	0.6312
Ours	No	No	**21.63**	**0.1200**	0.6375

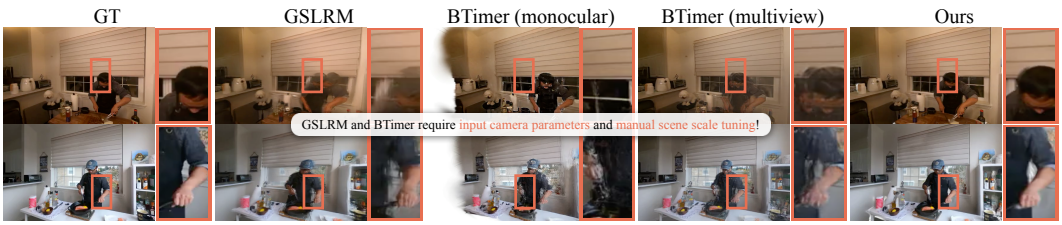

Figure 7: Qualitative comparison of our feedforward reconstruction model with the baselines on novel view renderings of dynamic scenes from the Neural3DVideo [94] dataset.

each timestep separately. BTimer [78] claims to support monocular video input but can also take a 4D grid. For fairness, we evaluate both versions: monocular and multi-view. The monocular version cannot generate views outside the input trajectory and struggles with large camera motions. As in static scenes, our method faces a harder task. It predicts both the Gaussians and the camera parameters for input/output views. Still, it produces sharper, more accurate results. Table 7 confirms this with better metrics, even when the baselines are manually tuned per scene. In this experiment, we use four views as input for each scene and the rest as targets. We also perform an ablation study by removing the camera token replacement and the temporal attention components, as shown in Table 8. The results indicate that both components contribute to improved quality in the rendered novel views.

5 Conclusion

We presented a two-stage framework for 4D video generation that leverages large-scale 4D video diffusion models and a feedforward reconstruction network to produce dynamic Gaussian splats from synchronized multi-view videos. While our method achieves state-of-the-art performance across multiple benchmarks, several **limitations** remain. First, the current design does not support full 360-degree scene generation. Second, although multi-view consistency is improved over prior methods, some layering artifacts can still appear in the reconstructed Gaussian splats. Third, inference remains computationally expensive, requiring approximately 4 minutes to generate 8 views and 29 frames on a single A100 GPU. Future work may explore model distillation to improve inference speed and expand scene coverage.

Table 8: Ablation of our feedforward reconstruction model on the test set of the dynamic Kubric [84] dataset.

Method	PSNR
w/o cam. token replacement	22.48
w/o temporal attention	22.60
Full model	**23.39**

Acknowledgement We extend our gratitude to Tuan Duc Ngo, Jiahao Luo, Hanwen Liang and Guochen Qian for their valuable assistance with data preparation and model training.

References

[1] Daniel Watson, Saurabh Saxena, Lala Li, Andrea Tagliasacchi, and David J Fleet. Controlling space and time with diffusion models. *ICLR*, 2025.

[2] Rundi Wu, Ruiqi Gao, Ben Poole, Alex Trevithick, Changxi Zheng, Jonathan T. Barron, and Aleksander Holynski. CAT4D: Create Anything in 4D with Multi-View Video Diffusion Models. 2024.

[3] Yuyang Zhao, Chung-Ching Lin, Kevin Lin, Zhiwen Yan, Linjie Li, Zhengyuan Yang, Jianfeng Wang, Gim Hee Lee, and Lijuan Wang. Genxd: Generating any 3d and 4d scenes. *ICLR*, 2025.

[4] Wenqiang Sun, Shuo Chen, Fangfu Liu, Zilong Chen, Yueqi Duan, Jun Zhang, and Yikai Wang. Dimensionx: Create any 3d and 4d scenes from a single image with controllable video diffusion. 2024.

[5] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas MÃ¼ller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control. In *CVPR*, 2025.

[6] Mark YU, Wenbo Hu, Jinbo Xing, and Ying Shan. Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models, 2025.

[7] 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. In *ECCV*, 2024.

[8] Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, and Di Zhang. Recammaster: Camera-controlled generative rendering from a single video, 2025.

[9] 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. 2024.

[10] Ruizhi Shao, Youxin Pang, Zerong Zheng, Jingxiang Sun, and Yebin Liu. Human4dit: Free-view human video generation with 4d diffusion transformer. 2024.

[11] Bing Li, Cheng Zheng, Wenxuan Zhu, Jinjie Mai, Biao Zhang, Peter Wonka, and Bernard Ghanem. Vivid-zoo: Multi-view video generation with diffusion model, 2024.

[12] Haiyu Zhang, Xinyuan Chen, Yaohui Wang, Xihui Liu, Yunhong Wang, and Yu Qiao. 4diffusion: Multi-view video diffusion model for 4d generation. 2024.

[13] Yiming Xie, Chun-Han Yao, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d: Dynamic 3d content generation with multi-frame and multi-view consistency. 2024.

[14] Jianhong Bai, Menghan Xia, Xintao Wang, Ziyang Yuan, Xiao Fu, Zuozhu Liu, Haoji Hu, Pengfei Wan, and Di Zhang. Syncammaster: Synchronizing multi-camera video generation from diverse viewpoints, 2024.

[15] Chaoyang Wang, Peiye Zhuang, Tuan Duc Ngo, Willi Menapace, Aliaksandr Siarohin, Michael Vasilkovsky, Ivan Skorokhodov, Sergey Tulyakov, Peter Wonka, and Hsin-Ying Lee. 4real-video: Learning generalizable photo-realistic 4d video diffusion, 2024.

[16] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. 2024.

[17] Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. 2025.

[18] Haoyang Huang, Guoqing Ma, Nan Duan, Xing Chen, Changyi Wan, Ranchen Ming, Tianyu Wang, Bo Wang, Zhiying Lu, Aojie Li, Xianfang Zeng, Xinhao Zhang, Gang Yu, Yuhe Yin, Qiling Wu, Wen Sun, Kang An, Xin Han, Deshan Sun, Wei Ji, Bizhu Huang, Brian Li, Chenfei Wu, Guanzhe Huang, Huixin Xiong, Jiaxin He, Jianchang Wu, Jianlong Yuan, Jie Wu, Jiashuai Liu, Junjing Guo, Kaijun Tan, Liangyu Chen, Qiaohui Chen, Ran Sun, Shanshan Yuan, Shengming Yin, Sitong Liu, Wei Chen, Yaqi Dai, Yuchu Luo, Zheng Ge, Zhisheng Guan, Xiaoniu Song, Yu Zhou, Binxing Jiao, Jiansheng Chen, Jing Li, Shuchang Zhou, Xiangyu Zhang, Yi Xiu, Yibo Zhu, Heung-Yeung Shum, and Daxin Jiang. Step-video-ti2v technical report: A state-of-the-art text-driven image-to-video generation model, 2025.

[19] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie gen: A cast of media foundation models, 2024.

[20] Veo-Team:, Agrim Gupta, Ali Razavi, Andeep Toor, Ankush Gupta, Dumitru Erhan, Eleni Shaw, Eric Lau, Frank Belletti, Gabe Barth-Maron, Gregory Shaw, Hakan Erdogan, Hakim Sidahmed, Henna Nandwani, Hernan Moraldo, Hyunjik Kim, Irina Blok, Jeff Donahue, José Lezama, Kory Mathewson, Kurtis David, Matthieu Kim Lorrain, Marc van Zee, Medhini Narasimhan, Miaosen Wang, Mohammad Babaeizadeh, Nelly Papalampidi, Nick Pezzotti, Nilpa Jha, Parker Barnes, Pieter-Jan Kindermans, Rachel Hornung, Ruben Villegas, Ryan Poplin, Salah Zaiem, Sander Dieleman, Sayna Ebrahimi, Scott Wisdom, Serena Zhang, Shlomi Fruchter, Signe Nørly, Weizhe Hua, Xinchen Yan, Yuqing Du, and Yutian Chen. Veo 2. 2024.

[21] 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*, 2021.

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ToG*, 2023.

[23] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025.

[24] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023.

[25] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *NeurIPS*, 2023.

[26] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 2023.

[27] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *ICCV*, 2023.

[28] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 2023.

[29] Joseph Zhu and Peiye Zhuang. Hifa: High-fidelity text-to-3d with advanced diffusion guidance. In *ICLR*, 2023.

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[31] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 2022.

[32] 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. In *ICCV*, 2023.

[33] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *ICLR*, 2024.

[34] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, et al. Text-to-4d dynamic scene generation. 2023.

[35] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. In *CVPR*, 2024.

[36] Yanqin Jiang, Li Zhang, Jin Gao, Weimin Hu, and Yao Yao. Consistent4d: Consistent 360 {\deg} dynamic object generation from monocular video. 2023.

[37] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. 2023.

[38] Yuyang Yin, Dejia Xu, Zhangyang Wang, Yao Zhao, and Yunchao Wei. 4dgen: Grounded 4d content generation with spatial-temporal consistency. 2023.

[39] Yuyang Zhao, Zhiwen Yan, Enze Xie, Lanqing Hong, Zhenguo Li, and Gim Hee Lee. Animate124: Animating one image to 4d dynamic scene. 2023.

[40] Heng Yu, Chaoyang Wang, Peiye Zhuang, Willi Menapace, Aliaksandr Siarohin, Junli Cao, Laszlo A Jeni, Sergey Tulyakov, and Hsin-Ying Lee. 4real: Towards photorealistic 4d scene generation via video diffusion models. In *NeurIPS*, 2024.

[41] Chaoyang Wang, Peiye Zhuang, Aliaksandr Siarohin, Junli Cao, Guocheng Qian, Hsin-Ying Lee, and Sergey Tulyakov. Diffusion priors for dynamic view synthesis from monocular videos, 2024.

[42] 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.

[43] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. 2022.

[44] Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter1: Open diffusion models for high-quality video generation. 2023.

[45] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023.

[46] OpenAI. Video generation models as world simulators, 2024.

[47] Willi Menapace, Aliaksandr Siarohin, Ivan Skorokhodov, Ekaterina Deyneka, Tsai-Shien Chen, Anil Kag, Yuwei Fang, Aleksei Stoliar, Elisa Ricci, Jian Ren, et al. Snap video: Scaled spatiotemporal transformers for text-to-video synthesis. In *CVPR*, 2024.

[48] 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.

[49] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. 2024.

[50] Shiyuan Yang, Liang Hou, Haibin Huang, Chongyang Ma, Pengfei Wan, Di Zhang, Xiaodong Chen, and Jing Liao. Direct-a-video: Customized video generation with user-directed camera movement and object motion. In *SIGGRAPH*, 2024.

[51] Dejia Xu, Weili Nie, Chao Liu, Sifei Liu, Jan Kautz, Zhangyang Wang, and Arash Vahdat. Camco: Camera-controllable 3d-consistent image-to-video generation. 2024.

[52] Qitai Wang, Lue Fan, Yuqi Wang, Yuntao Chen, and Zhaoxiang Zhang. Freevs: Generative view synthesis on free driving trajectory. 2024.

[53] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *ICCV*, 2021.

[54] Richard Tucker and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ToG*, 2018.

[55] Zhiwen Fan, Kairun Wen, Wenyan Cong, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. Instantsplat: Sparse-view gaussian splatting in seconds, 2024.

[56] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, 2021.

[57] Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3r: Gradient guided generalizable reconstruction. In *European Conference on Computer Vision*, 2024.

[58] Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a generalizable dynamic radiance field from monocular videos. In *ICCV*, 2023.

[59] Wenyan Cong, Hanxue Liang, Peihao Wang, Zhiwen Fan, Tianlong Chen, Mukund Varma, Yi Wang, and Zhangyang Wang. Enhancing neRF akin to enhancing LLMs: Generalizable neRF transformer with mixture-of-view-experts. In *ICCV*, 2023.

[60] Mukund Varma T, Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Is attention all that neRF needs? In *ICLR*, 2023.

[61] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *ICLR*, 2024.

[62] David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024.

[63] 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. *ECCV*, 2024.

[64] Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. Mvsplat360: Feed-forward 360 scene synthesis from sparse views. 2024.

[65] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *ECCV*, 2024.

[66] Xuanchi Ren, Yifan Lu, Hanxue Liang, Jay Zhangjie Wu, Huan Ling, Mike Chen, Francis Fidler, Sanja annd Williams, and Jiahui Huang. Scube: Instant large-scale scene reconstruction using voxsplats. In *NeurIPS*, 2024.

[67] 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. In *ICLR*, 2025.

[68] Hanwen Jiang, Hao Tan, Peng Wang, Haian Jin, Yue Zhao, Sai Bi, Kai Zhang, Fujun Luan, Kalyan Sunkavalli, Qixing Huang, and Georgios Pavlakos. Rayzer: A self-supervised large view synthesis model. 2025.

[69] Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs. 2024.

[70] Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. *CVPR*, 2025.

[71] Botao Ye, Sifei Liu, Haofei Xu, Li Xueting, Marc Pollefeys, Ming-Hsuan Yang, and Peng Songyou. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *ICLR*, 2025.

[72] Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. Pf3plat: Pose-free feed-forward 3d gaussian splatting. *ICML*, 2025.

[73] Xiaoming Zhao, Alex Colburn, Fangchang Ma, Miguel Ángel Bautista, Joshua M. Susskind, and Alexander G. Schwing. Pseudo-Generalized Dynamic View Synthesis from a Video. In *ICLR*, 2024.

[74] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. *ICLR*, 2025.

[75] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024.

[76] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with mast3r, 2024.

[77] Jiawei Ren, Kevin Xie, Ashkan Mirzaei, Hanxue Liang, Xiaohui Zeng, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, and Huan Ling. L4gm: Large 4d gaussian reconstruction model. In *NeurIPS*, 2024.

[78] Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, and Jiahui Huang. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. 2024.

[79] William Peebles and Saining Xie. Scalable diffusion models with transformers. 2022.

[80] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda,

Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical ai, 2025.

[81] Chun-Han Yao, Yiming Xie, Vikram Voleti, Huaizu Jiang, and Varun Jampani. Sv4d 2.0: Enhancing spatio-temporal consistency in multi-view video diffusion for high-quality 4d generation, 2025.

[82] Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels, 2024.

[83] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. 2022.

[84] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. 2022.

[85] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *CVPR*, 2020.

[86] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[87] 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 *CVPR*, 2024.

[88] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Tianyou Liang, Guanying Chen, Shuguang Cui, and Xiaoguang Han. Mvimgnet: A large-scale dataset of multi-view images. In *CVPR*, 2023.

[89] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *ICCV*, 2021.

[90] Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, Yaohui Wang, Xinyuan Chen, Ying-Cong Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench++: Comprehensive and versatile benchmark suite for video generative models. 2024.

[91] Mohammad Asim, Christopher Wewer, Thomas Wimmer, Bernt Schiele, and Jan Eric Lenssen. Met3r: Measuring multi-view consistency in generated images. In *CVPR*, 2024.

[92] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ToG*, 2017.

[93] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ToG*, 2019.

[94] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *CVPR*, 2022.

[95] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, June 2024.

Appendix

A Visual results for comparing architectures for 4D video generation

Figure S1 presents a qualitative comparison of outputs from various model architectures: *SV4D* [13], *sequential*, *parallel*, and our proposed *fused view-time attention* architecture. Each example showcases a frame from a 4D video. The *fused view-time attention* model produces the most consistent and realistic results, closely resembling the ground truth in both shape and appearance. In contrast, the *sequential* architecture exhibits lighting artifacts and fails to maintain a clean background, particularly in the Objaverse scenes. The *parallel* architecture performs better but still shows noticeable temporal instability and degradation in fine details. *SV4D* suffers from significant blurriness and structural distortions, underscoring the advantages of joint view-time modeling in our proposed approach. Please refer to Table 2 from the main paper for a quantitative comparison. The results for the sequential and parallel architectures stem from our own reimplementation of these architectures, so that all architectures use the same video model as backbone for a fair comparison (besides SV4D).

B Additional details on 4D video diffusion model

Architecture details. The base video model is a latent diffusion model built on a DiT backbone, consisting of 32 DiT blocks with a hidden size of 4096, and a total of 11B learnable parameters. We use rotary positional embedding (RoPE) for its relative encoding properties and strong generalization across varying resolutions and durations. The model employs a convolutional autoencoder similar to that in CogVideo [16], achieving $8\times$ compression in the spatial dimensions and $4\times$ in the temporal dimension. We fine-tune our 4D model using videos at a resolution of 144×256, and observe that it generalizes well to higher resolutions (e.g., 288×512) and longer durations without additional training.

Training data composition. Training Data Composition. Our training set comprises a combination of synthetic 4D data from Objaverse and Kubric, 2D transformed videos, and videos of static scenes. Each training batch consists of 40% Objaverse data, 20% Kubric, 20% 2D transformed videos, and 20% static scene videos. For the static scenes, we duplicate and stack frames to construct a 4D video structure, although no actual object motion is present. To prevent the model from learning a trivial solution that simply replicates the first frame across all views, we find it necessary to remove frame conditioning when using freeze-time videos. Otherwise, the model tends to ignore viewpoint variation and fails to capture meaningful temporal dynamics. In addition, we observe that randomly reversing the order of viewpoints serves as an effective augmentation strategy that improves the model's generalization capability.

Training details. Training Setup. We train the model on 48 A100 GPUs with a batch size of 96, using sequences of 8 views and 29 frames. The learning rate is set to $1e-4$ with a warm-up schedule. The model converges quickly and begins producing plausible results after approximately 2000 iterations. We switch to fine-tuning the model on sequences with 8 views and 61 frames at 4000 iterations and the finetuning continues for an additional 2000 iterations. This costs around 2 days, or roughly 96 GPU days. We observe that the trained model generalizes well to sequences with varying numbers of frames, even when they differ from the configuration used during training.

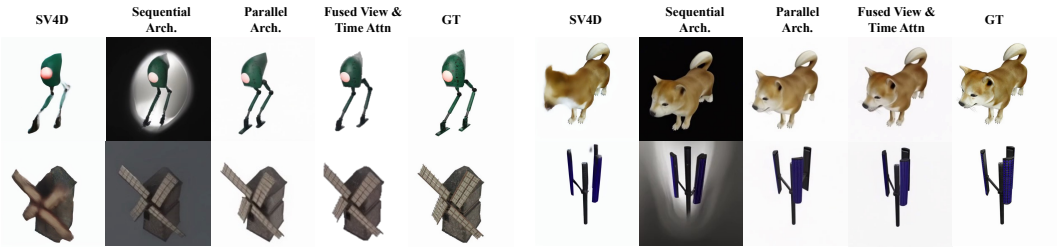

Figure S1: Qualitative comparison of the outputs of our proposed architecture (fused view-time attention) with our implementation of sequential and parallel architectures and SV4D [13].

Table S1: Compare with learnable view positional embedding for multi-view video generation.

Positional Embedding	Objaverse PSNR↑	Objaverse SSIM↑	Objaverse LPIPS↓	NVidia Dynamic Dataset PSNR↑	NVidia Dynamic Dataset SSIM↑	NVidia Dynamic Dataset LPIPS↓
Learnable View PE	19.50	0.899	0.161	**23.45**	0.751	**0.113**
$((v, t, x, y) \rightarrow (vT_{\max} + t, x, y))$	**22.49**	**0.909**	**0.113**	23.15	**0.752**	0.142

Sampling Strategy and Classifier-Free Guidance. We adopt a rectified flow sampler consistent with our base video model. In the setting where both freeze-time and fixed-view videos are provided as input, we find that classifier-free guidance (CFG) yields marginal improvements in output quality. Under this configuration, our model is capable of generating high-quality results with a small number of diffusion steps—for example, as shown in Tab. S2, using only 4 steps already produces temporally consistent outputs, particularly in background regions. Further refinement of the foreground, especially in areas with larger motion, occurs with additional steps. This suggests that our model could potentially benefit from distillation techniques aimed at reducing the number of inference steps.

However, when only a single video is used as input, CFG remains essential. In this case, the model relies more heavily on the input text to resolve ambiguities during generation.

Other variants of positional embedding for 4D video In addition to the design proposed in the main paper, we explored alternative formulations for converting 4D coordinates into 3D positional embeddings. Notably, we experimented with the transformation $(v, t, x, y) \rightarrow (v + t, x, y)$, based on the intuition that temporal indices are consecutive across rows and columns of the frame matrix. This mapping preserves the structural assumptions of the pretrained base video model.

Empirically, this variant performs comparably to our proposed embedding scheme when both freeze-time and fixed-view videos are used as input. However, when only one of the two input types is provided, the results become less stable. We attribute this to the ambiguity introduced by the $(v + t, x, y)$ formulation, which leads to duplicated or symmetric positions in the frame grid. Specifically, positions become indistinguishable along the diagonal of the view-time plane, making it difficult for the model to differentiate between the temporal and view dimensions. As a result, the model must rely more heavily on the input frames themselves to infer the underlying structure.

Comparing to learnable view positional embedding. We implemented a baseline that incorporates a learnable positional embedding for the view dimension. Specifically, for each token indexed by (v, t, x, y), we compute separate positional embeddings for the view v and the spatiotemporal coordinates (t, x, y). For the view dimension, we apply sinusoidal encoding with 16 learnable frequencies, followed by a two-layer MLP to produce the final view embedding. For (t, x, y), we adopt the 3D RoPE (Rotary Positional Embedding) used in the original base video model. To integrate both positional embeddings into the attention mechanism, we first add the view embedding to the key and query vectors, and then apply the 3D RoPE. This ensures that the RoPE is applied correctly while still incorporating view-specific information.

We compare this baseline against our proposed embedding under identical training and testing settings (see Tab. S1). We observe that the model with learnable view embeddings converges more slowly during training. While it achieves comparable performance on dynamic-scene datasets such as the NVIDIA dataset, it fails to produce correct backgrounds (e.g., white backgrounds in the Objaverse dataset) resulting in catastrophic failures. We hypothesize that this may be due to confusion arising from the mixed nature of the training data. Further investigation, such as tuning the training schedule or dataset composition, may help address this issue. In conclusion, the proposed embedding is empirically more stable to train, achieves competitive or superior quality, and introduces no additional parameters.

C Additional details on the feedforward reconstruction model

Training details Static and dynamic training use batch sizes of 14 and 1, respectively, and learning rates of 0.0002 and 0.00002. We sample uniformly across datasets in both stages. Static training runs for $20K$ iterations, and dynamic training runs for $15K$ iterations. The training is done on 32 A100 GPUs for around a day. We use the same hyperparameters for temporal attention as for global attention in VGGT [23]. The same hyperparameters as VGGT's depth head are also used for the

Table S2: Cross-view consistency and Cross-time quality assement for generation with different diffusion steps. Runtime is estimated for generating 4D videos with 8 views and 61 timestamps, in total 488 frames.

		Cross-View	Cross-Time (VBench [90])				
# Step	Time (s)	Met3R↓ [91]	Flickering↑	Motion↑	Subject↑	Background ↑	Image↑
4	47.2	0.187	94.6	97.8	96.3	97.7	64.7
8	89.4	0.184	94.5	97.7	96.5	97.7	65.6
16	173.8	0.183	94.4	97.7	96.6	97.7	65.7
40	472.0	0.173	99.1	99.5	97.7	98.4	66.2

Gaussian head, except the output dimension is set to 14: 3 for position refinement, 1 for opacity, 3 for scales, 4 for rotation (quaternion), and 3 for color offsets. Color and pose offsets are added following Splatt3r [69].

Computational efficiency. Table S3 provides an overview of the time and GPU memory usage required to run our feedforward reconstruction model on both dynamic and static datasets. Our model is capable of producing Gaussians for static and dynamic scenes within seconds. These metrics are calculated on an Nvidia A100 GPU. This experiment is conducted using inputs with a resolution of 350×518, following the standard input dimensions of VGGT.

Visual results for static scene novel view synthesis Figure S2 supports the quantitative results in Table 6 from the main paper. We compare our method with GSLRM [65] and BTimer [78] on LLFF [93] and Tanks & Temples [92] scenes. The baselines need ground-truth camera poses and a per-scene scale search, while our method predicts all camera parameters and uses no manual tuning. GSLRM and BTimer are trained with a photometric loss only, so their per-view Gaussians do not stay aligned when the input set grows. With 16 input views the misalignment causes layering artifacts on fine details, such as the fern leaves, and on thin parts like the back leg of the *Horse* statue. Our model avoids these artifacts, matching the gains in PSNR, SSIM, and LPIPS reported in the table. In Fig S3, we also compare our method to PixelSplat [62] and MVSplat [63] on the RealEstate10K [54] dataset. Our method produces visuals that more closely match the ground truth. Note that PixelSplat and MVSplat are trained specifically on RealEstate10K, so we compare on this dataset for fairness.

The difference between our renderings and those from the baselines is especially apparent when the target camera differs significantly from the input trajectory. Figure S4 shows renderings from our model compared to the baselines when the camera is moved backward and far from the set of input frames. Notably, our model is much better suited for view extrapolation, in part because it incorporates superior geometric priors, whereas the baselines rely solely on photometric losses.

Dataset details Table S4 provides an overview of the datasets used to train our models, summarizing key characteristics such as the presence of dynamic content, the type of content (object-centric or scene-level), the domain (real or synthetic), the approximate number of scenes, and the associated licenses. These datasets span a range of scenarios and content types, offering a diverse foundation for training models in our experiments.

Training data sensitivity analysis. Table S5 evaluates how training data affects the feedforward reconstruction model. The original model is trained on five datasets: RealEstate10K, DL3DV, MVImageNet, ACID, and static cuts of Kubric. The accompanying table shows performance when the model is trained on each dataset individually. For Neural3DVideo, we fine-tune a dynamic variant derived from each baseline. Training on all datasets results in a noticeable improvement. With more

Table S3: Runtime (seconds) and peak GPU memory (GBs) required by our feed-forward reconstruction network during inference on static and dynamic scene sequences, reported for varying numbers of input camera views and timesteps. *OOM* means the model has ran out of memory.

	1 Timestep (Static)		4 Timesteps		8 Timesteps		16 Timesteps	
# Input Views	Time (s)	Mem. (GB)	Time (s)	Mem. (GB)	Time (s)	Mem. (GB)	Time (s)	Mem. (GB)
2	0.1779	7.204	0.4313	10.282	0.6850	13.709	1.2317	20.571
4	0.2044	7.885	0.6467	13.192	1.1712	18.528	2.1725	32.213
8	0.2742	9.319	1.3009	18.933	2.4204	31.011	5.6977	60.172
16	0.5566	10.817	2.7396	24.989	5.2527	43.123	*OOM*	*OOM*

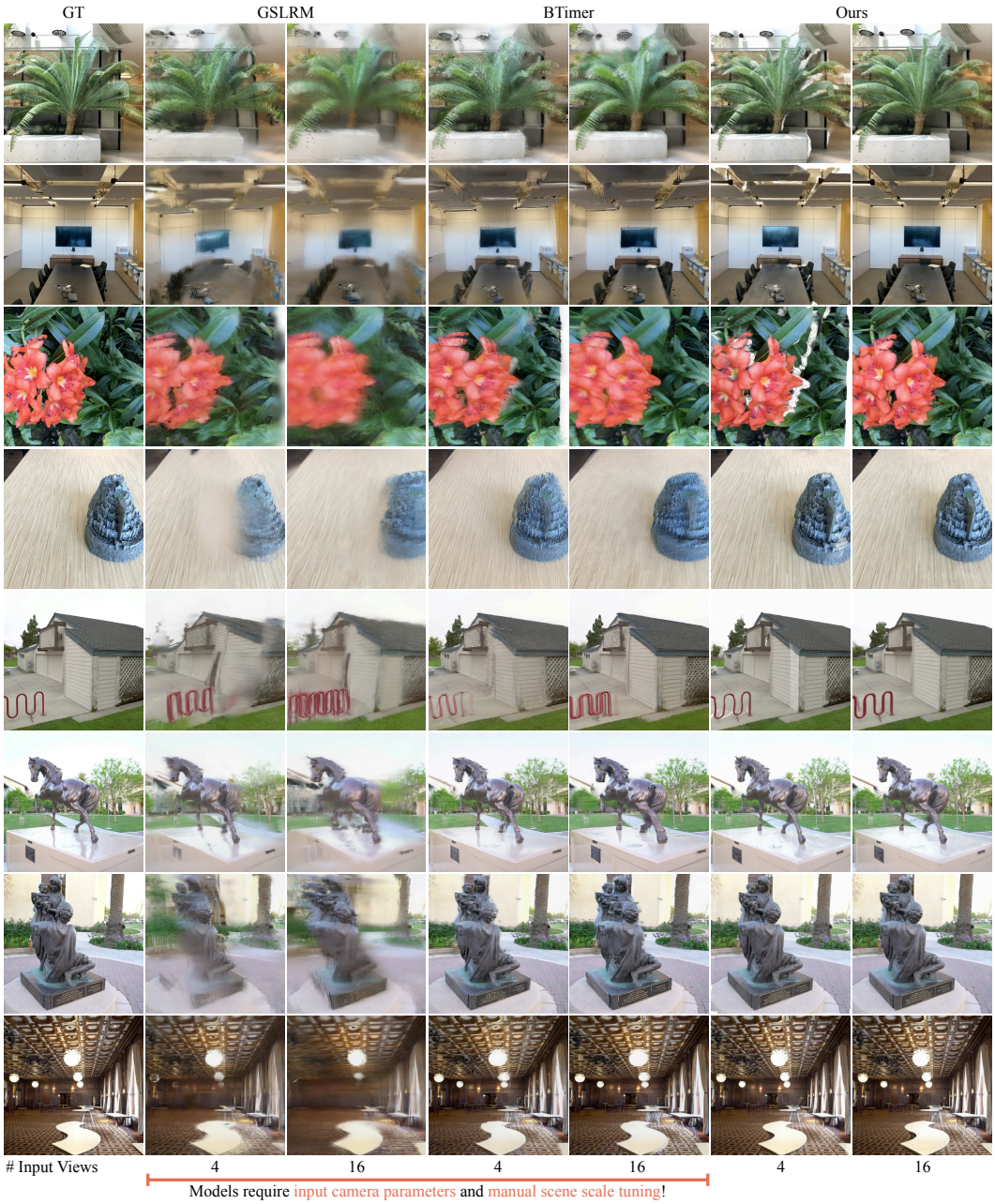

Figure S2: Qualitative comparison of our renderings with the baselines GSLRM [65] and BTimer [78] on the task of novel view synthesis for static scenes. Each method includes two variations, using 4 and 16 input views. Note that all variations of GSLRM and BTimer require input camera parameters and manual scene scale tuning.

Figure S3: Qualitative comparison of our renderings with the baselines PixelSplat [62] and MVSPlat [63] on the task of novel view synthesis for static scenes. Note that the baselines require input camera parameters, whereas our method infers the camera parameters from the input images.

Figure S4: Qualitative comparison of our results with the baselines for novel view synthesis of static scenes, where the target camera deviates significantly from the input trajectory.

computing power, one could possibly search over all possible subsets of these datasets to find the best possible results.

System diagnosis of the two stage generation pipeline. Finally, in Table S6 we present a quantitative diagnosis of the full pipeline on the Nvidia Dynamic Scene dataset, holding out views 1, 4, 7, and 10 at every timestep for evaluation. The optimization-based baseline, 4DGS [95], is run on the first view at all timesteps plus the first timestep of every view (first row and first col of the 4D grid); in a second setting, we first complete the grid with the 4D video generator (while using the first column and the first row as input) and then optimize with 4DGS; in a third setting, we feed the generated grid to our feedforward reconstruction model, thereby applying the complete two-stage pipeline. Our pipeline outperforms the baselines in quality and reduces reconstruction time from over an hour (4DGS) to under 3 seconds. Even if this or another optimization-based method could produce better visual results, it would not be an option due to their prohibitive runtime for every scene. To avoid potential issues with baseline depth/optical-flow priors failing on generated content, we utilized 4DGS as the standard prior-free method for dynamic scene reconstruction.

Table S4: Specification and licenses for the datasets used to train our models.

Dataset	**Dynamic**	**Content**	**Domain**	**# Scenes**	**License**
RealEstate10K [54]		*Scene*	Real	80K	CC-BY (per video)
MVImageNet [88]		*Object*	Real	220K	Custom (password-protected)
DL3DV [87]		*Scene*	Real	10K	NonCommercial (custom terms)
Kubric [84]	✓	*Object+Scene*	Synthetic	3K	Apache 2.0
Dynamic Objaverse [45]	✓	*Object*	Synthetic		ODC-By v1.0 (mixed per object)

Table S5: Analysis of training data sensitivity in feedforward reconstruction.

Train Dataset	Eval Dataset	Neural3DVideo (Dynamic)	Tanks & Temples (Static)	LLFF (Static)
RealEstate10K		20.11	20.37	18.03
DL3DV		20.17	20.48	18.41
MVImageNet		19.97	20.03	18.33
ACID		18.31	18.28	17.23
Kubric		17.14	17.73	17.16
All		**21.63**	**20.85**	**18.95**

Table S6: System diagnosis of the two stage generation pipeline.

	4DGS [95]	MV-Video Gen. + 4DGS	MV-Video Gen. + Feedforward Recon.
PSNR	17.60	19.05	**19.14**

Broader impact By supporting 4D content creation, our method opens new possibilities in animation and visual effects. Nonetheless, careful consideration is required to prevent its exploitation for deceptive or harmful purposes, such as identity forgery.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: we clearly position our paper among the related work and provide results to support our claims and contributions.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: yes, we provide a discussion on the limitations of our work in the conclusion section.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: the paper does not include theoretical results.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: we provide implementation details necessary to reproduce our results in the main paper and the supplement.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: code to be released upon internal approval.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: we provide the details about our experiments in the main paper and the supplement.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: repeating experiments with large models is too computationally expensive. As a result, it is not standard in the literature to accompany the results with error bars.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

Justification: we provide details in the paper and the supplement.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: we hereby confirm that our research provided in the paper conforms with the NeurIPS code of ethics in every respect.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: yes, we discuss it in a dedicated section in the supplement.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: the paper poses no such risk. We'll follow the standard safeguard practices upon release.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: all the assets are properly cited and detailed in the paper or the supplement.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: the paper does not release new assets.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: our user study and its details are found in the supplement.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: we provide use study details in the supplement, which does not include any potential risks.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: our core method does not involve LLMs as any important, original, or non-standard component.