

Appendix: Fused View-Time Attention and Feedforward Reconstruction for 4D Scene Generation

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 additional 2000 iterations. We observe that the trained model generalizes

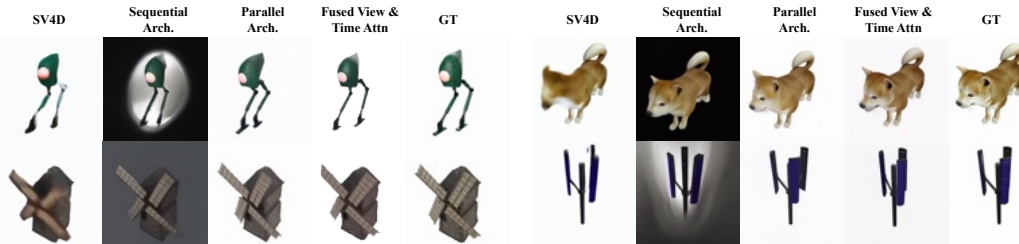


Figure S1: Qualitative comparison of the outputs of our proposed architecture (fused view-time attention) with our implementation of sequential and parallel architectures and SV4D [13].

well to sequences with varying numbers of frames, even when they differ from the configuration used during training.

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) is unnecessary, as it does not yield noticeable 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.

C Additional details on the feedforward reconstruction model

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. 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 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 [70].

Table S1 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.

Table S1: 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.

# Input Views	1 Timestep (Static)		4 Timesteps		8 Timesteps		16 Timesteps	
	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	<i>OOM</i>	<i>OOM</i>

# Step	Time (s)	Cross-View	Cross-Time (VBench [91])				
		Met3R↓ [92]	Flickering↑	Motion↑	Subject↑	Background ↑	Image↑
4	47.2	0.187	94.6	97.8	96.3	97.7	64.7
8	89.4	18.4	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

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.

Table S3: Specification and licenses for the datasets used to train our models.

Dataset	Dynamic	Content	Domain	# Scenes	License
RealEstate10K [55]		<i>Scene</i>	Real	80K	CC-BY (per video)
MVImageNet [89]		<i>Object</i>	Real	220K	Custom (password-protected)
DL3DV [88]		<i>Scene</i>	Real	10K	NonCommercial (custom terms)
Kubric [85]	✓	<i>Object+Scene</i>	Synthetic	3K	Apache 2.0
Dynamic Objaverse [45]	✓	<i>Object</i>	Synthetic		ODC-By v1.0 (mixed per object)

D Visual results for static scene novel view synthesis

Figure S2 supports the quantitative results in Table 4 from the main paper. We compare our method with GSLRM [66] and BTimer [79] on LLFF [94] and Tanks & Temples [93] 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 [63] and MVSplat [64] on the RealEstate10K [55] 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.

E Dataset details

Table S3 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.

F 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.

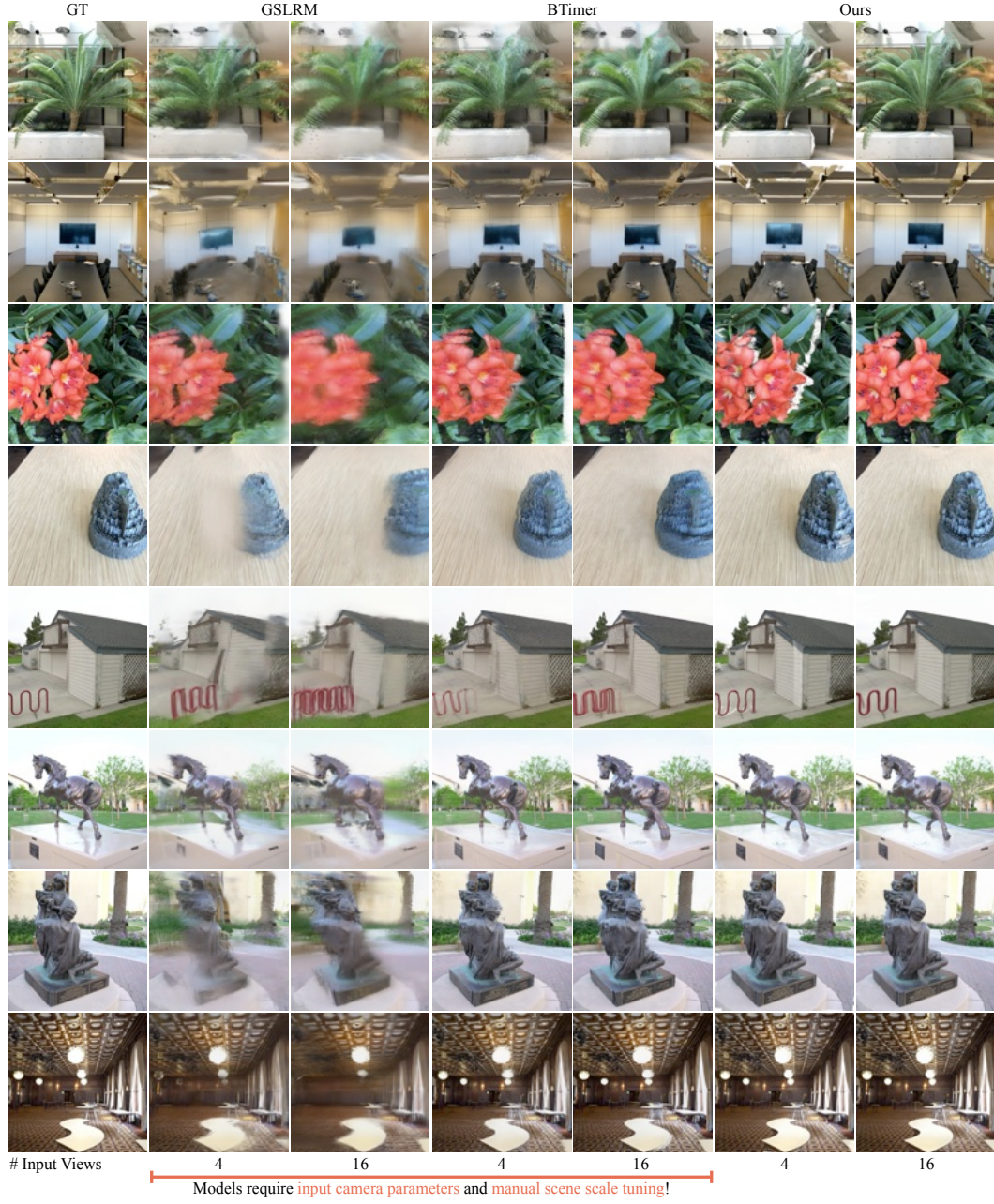


Figure S2: Qualitative comparison of our renderings with the baselines GSLRM [66] and BTimer [79] on the task of novel view synthesis for static scenes. Each method includes two variations, using 4 and 16 input views. Note that all variations of GSLRM and BTimer require **input camera parameters** and **manual scene scale tuning**.



Figure S3: Qualitative comparison of our renderings with the baselines PixelSplat [63] and MVS-Plat [64] 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.

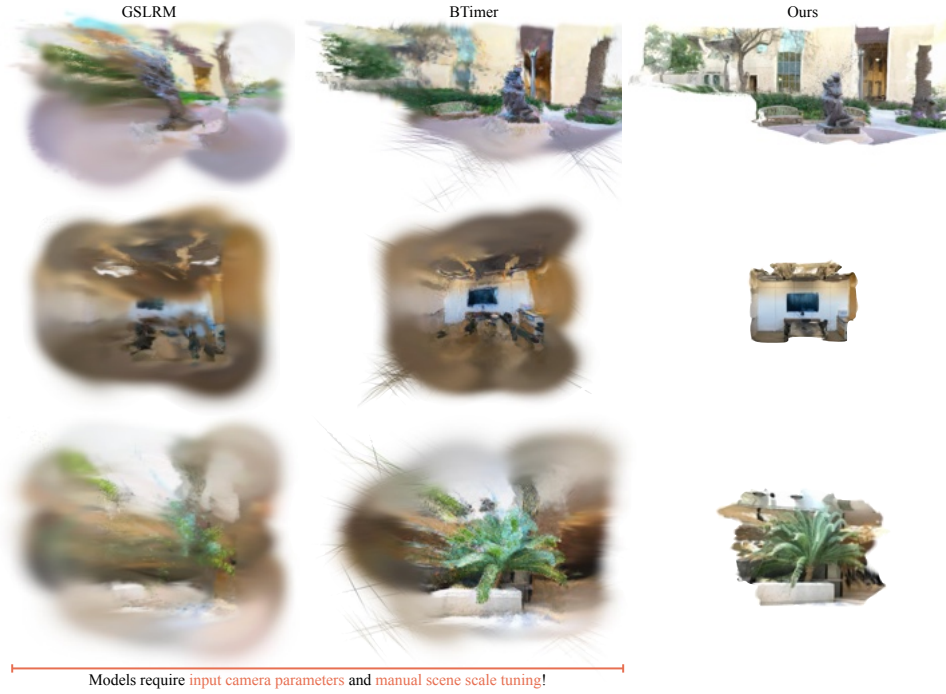


Figure S4: Qualitative comparison of our results with the baselines for novel view synthesis of static scenes, where the target camera deviates significantly from the input trajectory.