Efficient Camera-Controlled Video Generation of Static Scenes via Sparse Diffusion and 3D Rendering

Jieying Chen Jeffrey Hu Joan Lasenby Ayush Tewari
University of Cambridge

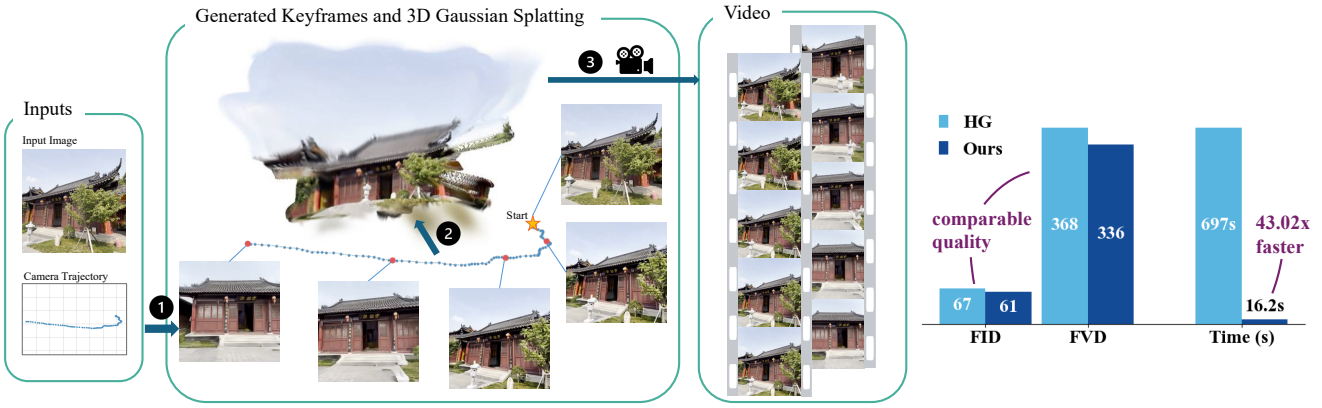

Figure 1. **Teaser.** Left: Overview of our approach. Given an input image and a camera trajectory, SRENDER generates sparse keyframes, reconstructs the 3D scene, and renders the full video efficiently. Right: On average, our method is 43.02 times faster than the history-guided video diffusion baseline (HG) [42] when generating 20-second 30-fps videos from the DL3DV dataset, achieving real-time performance while maintaining comparable or better video quality.

Abstract

Modern video generative models based on diffusion models can produce very realistic clips, but they are computationally inefficient, often requiring minutes of GPU time for just a few seconds of video. This inefficiency poses a critical barrier to deploying generative video in applications that require real-time interactions, such as embodied AI and VR/AR. This paper explores a new strategy for camera-conditioned video generation of static scenes: using diffusion-based generative models to generate a sparse set of keyframes, and then synthesizing the full video through 3D reconstruction and rendering. By lifting keyframes into a 3D representation and rendering intermediate views, our approach amortizes the generation cost across hundreds of frames while enforcing geometric consistency. We further introduce a model that predicts the optimal number of keyframes for a given camera trajectory, allowing the system to adaptively allocate computation. Our final method, SRENDER, uses very sparse keyframes for simple trajectories and denser ones for complex camera motion. This results in video generation that is more than 40 times faster than the diffusion-based baseline in generating 20 seconds of video, while maintaining high visual fidelity and temporal stability, offering a practical path toward efficient and controllable video synthesis.

1. Introduction

Generative video models have recently achieved impressive visual fidelity. State-of-the-art models such as Sora [37] and Wan [47] can synthesize photorealistic content that rivals cinematic footage. However, these successes come at a cost: current advances use diffusion-based [17] or flow-matching-based [31] techniques to generate every frame and are extremely computationally expensive. Producing even a short ten-second sequence can require up to tens of thousands of large neural network evaluations due to iterative denoising in diffusion models, amounting to several minutes of GPU time on high-end hardware. This inefficiency prevents real-time use and significantly limits ap-

plications in embodied AI applications, interactive content creation, and AR/VR.

While some research has attempted to make these models more efficient by designing low-dimensional latent space architectures [29, 57], or by utilizing distillation methods [55], all existing methods still rely on neural networks to generate every frame of the video. In this paper, we challenge this paradigm. Videos are inherently redundant, as many frames depict the same underlying 3D scene under gradually varying conditions such as viewpoint, motion, or lighting. Our goal is to develop a video generation framework that explicitly leverages these redundancies to enable efficient video synthesis—a direction that, to the best of our knowledge, has not been explored before.

In this work, we focus on the practically relevant setting of camera-conditioned generation of static scenes. This allows us to study the core principle of leveraging scene redundancy for efficient video synthesis, without the additional complexity of object motion or deformation. Camera-controlled video generation is an active area of research with several recent advances [19, 42, 53]. However, as previously mentioned, all these advances rely on generating every frame with neural networks. Recent work has demonstrated that incorporating 3D priors into video generation improves temporal consistency and novel-view controllability [19, 53]. However, in all cases, 3D priors have only been used as an internal representation, or for auxiliary constraints. Every frame of the final video is still produced by diffusion-like neural generative models. Thus, while incorporating 3D priors has improved quality, it has not fundamentally addressed the efficiency barrier. This paper addresses this missing piece.

To this end, we present our method, SRENDER, which generates full videos by first synthesizing a sparse set of keyframes with diffusion models and then generating the dense video through 3D scene reconstruction and rendering. This approach leverages the inherent 3D structure of visual scenes: high-quality reconstructions can be obtained even from sparse multi-view observations, without requiring dense video input. The camera-controlled video can be rendered very efficiently from the reconstructed 3D scene using standard physically-based rendering techniques. We build on advances in 3D reconstruction and rendering, particularly 3D Gaussian Splatting (3DGS) [23] that can render complex photo-realistic scenes at very high framerates, and follow-ups [22, 35] that can reconstruct 3DGS from multi-view observations with neural networks.

SRENDER also includes a model that adaptively selects how many keyframes to generate for a given camera trajectory. Simple trajectories with smooth motion or limited parallax can be reconstructed accurately from very few keyframes, whereas complex trajectories with large viewpoint changes require denser sampling. By predicting this keyframe budget directly from the camera path, SRENDER allocates computation where it is most needed, maintaining visual fidelity while minimizing redundant generation.

Across videos up to twenty seconds long, our adaptive keyframe selection model chooses between 4 and 35 keyframes, corresponding to generating at most one-tenth as many frames as would be required by a standard 30 fps video. Our approach achieves more than 40× speed-up on DL3DV [30] and more than 20× speed-up on RealEstate10k [12], while maintaining comparable visual quality and temporal consistency. These results demonstrate that explicit 3D reasoning and adaptive sparse generation can dramatically reduce the computational cost of video synthesis without sacrificing fidelity.

2. Related Work

2.1. Camera-controlled Video Models

Video generative models have seen fast progress in recent years, driven by diffusion models [17]. Most models focus on text-to-video, or image-to-video tasks [2, 10, 13, 37, 47]. Following the success of latent diffusion models in the image domain, early video models trained 3D UNets in the latent space of 3D VAEs [2]. Subsequent models replaced the UNet with a transformer for its greater scability [10, 47]. Most of the video diffusion models follow the standard diffusion procedure, denoising all frames of the video jointly from the same noise level. A recent influential work, diffusion forcing, breaks from this trend by proposing to add independent noise levels to each frame [5], and diffuse frames with different noise levels together. This strategy enables different denoising strategies at inference time, including autoregressive decoding for long-range video generation with variable-length condition frames, as well as video interpolation between keyframes. This piece of work then inspired a flurry of work on autoregressive video models [1, 6–9, 20, 32, 58]. Camera-controlled video generation is a subfield of video generation that has been shown to work very well. Existing methods for this problem either directly encode the desired camera poses to condition the generative model [14, 60], or make explicit use of the 3D structure of the world, e.g., by computing a point cloud from input images and rendering it from the desired camera view to condition the generative model [19, 40]. In this work, we focus specifically on the camera-controlled video generation problem, and we opt for a diffusion-forcing architecture. This design choice enables the generation of sparse keyframes with strong scene consistency and flexible conditioning on previous frames.

In general, inference with video diffusion models is very expensive. Even generating short (<3s) clips can take minutes on high-end hardware [1, 18, 47, 54]. Several attempts have been made at making these models faster, e.g., by us-

ing teacher-student distillation [55] or caching [25, 28, 52, 55, 62]. However, all approaches rely on neural networks to generate every frame of the video, and do not take advantage of the information redundancies in the video signal. Our method is complementary to these advances. Any improvement in diffusion inference directly reduces the cost of generating sparse keyframes, while the overall efficiency gains of our method stem from not generating intermediate frames with neural networks at all.

2.2. 3D Reconstruction

Advances in 3D representations, particularly 3D Gaussian Splatting (3DGS) [23], have enabled high-quality reconstruction of scenes from collections of images. Many recent models train neural networks to regress 3DGS representations directly from posed input images [3, 44]. These approaches can recover detailed and consistent 3D geometry without any test-time optimization. More recent systems [22] extend this capability by leveraging architectures inspired by DUST3R [48, 51], enabling efficient 3D reconstruction even from unposed image sets [22, 35].

Most progress in this area has focused on *deterministic* reconstruction. Such models reconstruct only the parts of the scene directly visible in the input images and cannot represent the full distribution of plausible 3D scenes consistent with the observations. As a result, although they produce high-quality geometry, they cannot be used as generative models. Some works have explored *generative* 3D reconstruction. Early approaches [11, 33] optimize a 3D representation using image diffusion models, but these optimization-based procedures are slow and typically limited to object-level scenes. More recent methods integrate diffusion and 3D representations [43, 45], enabling partially feed-forward 3D generation. However, these approaches have not yet achieved the visual quality, stability, or multi-view consistency seen in state-of-the-art video diffusion models. Closely related are methods that first use image diffusion models to generate multi-view images from a single input image and then fit a 3D representation [34, 36]. While these methods can produce coherent 3D assets, they remain restricted to object-centric settings and do not scale to full scenes or long camera trajectories. Recently, there have also been works that combine the generative capability of video diffusion models with an explicit 3D scene representation [19, 24, 56], achieving impressive results for generative 3D scene reconstruction.

Our method replaces dense video frame generation in the video model with a deterministic feed-forward 3D reconstruction model that reconstructs a 3DGS representation from the generated sparse keyframes. The video generation speed can thus be greatly improved as the deterministic feed-forward 3D reconstruction and rendering of the intermediate frames are much faster than diffusion-based frame synthesis.

Algorithm 1 Keyframe Selection

Require: Frames $\mathcal{F} = \{F_1, F_2, \ldots, F_N\}$, camera poses $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$, coverage threshold τ

Initialization:
Point Cloud Generation:
Obtain global point cloud $C_{global} = \text{VGGT}(\mathcal{F})$, consisting of sub-point clouds $\{C_1, C_2, \ldots, C_N\}$ corresponding to each frame.
Initialize selected frame set $\mathcal{S} \leftarrow \{F_1\}$ and combined point cloud $C \leftarrow C_1$

Iterative Selection:
for each subsequent frame $F_i \in \mathcal{F} \setminus \mathcal{S}$ **do**
    Project current point cloud C onto image plane using pose P_i
    Compute coverage ratio r_i of projected points on F_i
    if $r_i < \tau$ **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{F_i\}$
        $C \leftarrow C \cup C_i$
    end if
end for
return selected frame set $\mathcal{S}$

3. Keyframe Diffusion and 3D Rendering

Figure 2 provides an overview of SRENDER. Given an input image and a specified camera trajectory, the goal is to generate a video sequence that starts from the input image and corresponds to that trajectory. Rather than generating every frame through diffusion, SRENDER synthesizes a sparse set of *keyframes* and renders the remaining frames efficiently via 3D reconstruction.

We begin by using a *keyframe density predictor* that analyzes the camera trajectory and determines the optimal sparsity of keyframes. This model adaptively allocates computation based on motion complexity. Next, a diffusion-based *keyframe generator* synthesizes the selected keyframes conditioned on the input image and camera poses corresponding to the keyframes. The resulting multi-view set captures the appearance of the static scene from diverse viewpoints along the conditioned camera trajectory. We then reconstruct a 3D Gaussian representation from these keyframes using a deterministic (non-generative) neural network. Once reconstructed, a dense and geometrically consistent video can be rendered at high frame rates.

3.1. Adaptive Keyframe Selection

A central design choice in SRENDER is determining how many keyframes to generate for a given camera trajectory. Dense sampling increases computational cost, while overly

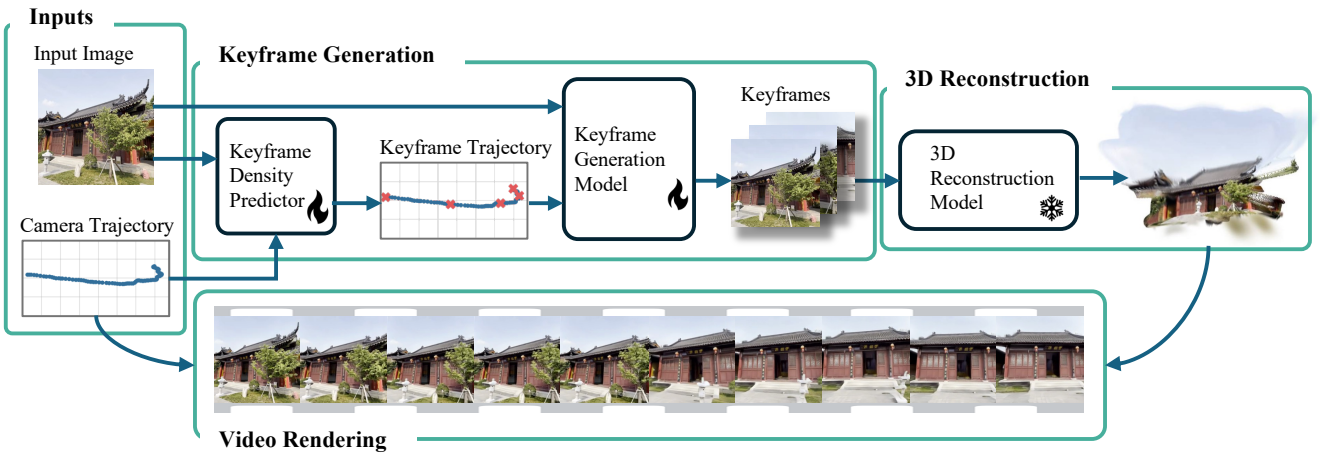

Figure 2. **Overview of SRENDER.** Given an image and target camera trajectory, a keyframe density predictor predicts the optimal keyframe density for the depicted scene and camera trajectory. Keyframe poses are then uniformly sampled along the trajectory before being fed to the keyframe generation model together with the input image for generating keyframes. A 3D reconstruction model takes the keyframes and generates the 3D representation of the scene. Finally, the video is rendered from the 3D scene along the input camera trajectory.

sparse sampling leads to incomplete 3D reconstructions and visible holes in rendered views.

The optimal keyframe set depends jointly on the camera path and the underlying scene geometry. Smooth trajectories or limited parallax may require only a few keyframes, whereas large viewpoint changes or complex geometry demand denser sampling. We formulate this as a learning problem in which the optimal number of keyframes is predicted from the camera trajectory and scene appearance.

Model. We train a transformer-based keyframe density predictor. The model takes the full sequence of camera poses as input, each represented as an independent token. Scene appearance is incorporated by extracting the global feature token with a DINOv2 [38] image encoder and appending it as an additional token. The tokens are processed by multiple self-attention blocks, and the resulting features are averaged and passed through a lightweight MLP to predict the optimal number of keyframes.

Supervision. Ground-truth keyframe densities are derived automatically from the RealEstate10k [12] and DL3DV [30] datasets. For each video, we reconstruct a point cloud using VGGT [48]. Starting from the first frame, we iteratively project the collective point cloud of the selected keyframes onto subsequent frames. When the projected coverage falls below a threshold, we mark the current frame as a new keyframe. We provide a pseudo-algorithm in Algorithm 1. This approach ensures that the selected keyframes collectively cover all pixels across the video. The transformer is trained to regress the number of keyframes produced by this procedure.

Discussion. This adaptive mechanism allocates generation effort where it is most beneficial: sparse keyframes are generated for smooth or low-parallax trajectories, and denser ones for complex motion. After the keyframe count is obtained, the keyframe camera poses are sampled uniformly along the camera trajectory.

3.2. Keyframe Diffusion

Once the keyframe density predictor selects the target poses, we synthesize the corresponding keyframes using a diffusion-based generator. The goal is to produce a small set of high-quality, geometrically consistent views that capture the static scene from the specified viewpoints.

Our keyframe generator builds on recent advances in camera-conditioned video diffusion, particularly diffusion forcing [4] and history-guided video diffusion [42], which improve temporal and geometric coherence across frames.

3.2.1. Preliminaries

In a standard denoising diffusion model [17], a data sample $\mathbf{x}_0$ is gradually perturbed through a forward noising process $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, and a neural network learns to reverse this process by predicting the noise at each step. Generation begins by sampling $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ and applying

$$\mathbf{x}_{t-1} = f_\theta(\mathbf{x}_t, t, c),$$

where c denotes conditioning information such as camera pose. In practice, the denoising is performed by a neural network.

Diffusion Forcing and History Guidance. In video diffusion, $\mathbf{x}_0$ is a video composed of multiple image frames.

Each frame of the output video needs to be denoised. Diffusion Forcing [4] assigns an independent noise level to each frame, allowing the model to denoise frames with different noise levels together, thus possible to mix clean history frames with partially noised to-be-generated ones at arbitrary frame positions. Building on this, History-Guided Video Diffusion [42] applies classifier-free guidance [16] on subsets of frames (e.g., previously denoised frames) to encourage long-range consistency.

3.2.2. Our Model

We treat the selected keyframes as a very low-frame-rate video and train a history-guided diffusion model to capture their joint distribution. Conditioning is provided by the first input frame and the camera trajectory. The first frame serves as a stable appearance anchor for all generated keyframes.

However, directly training a diffusion model at extremely low frame rates is unstable: the large viewpoint jumps cause geometric and photometric drift. To address this, we adopt a *progressive training strategy*. We first train the model on high-frame-rate videos with dense supervision, then gradually reduce the effective frame rate by subsampling frames until it matches the sparse keyframe spacing used at inference. This process enables the model to learn short-range correspondences before handling large viewpoint changes, yielding stable and coherent multi-view generation even at high sparsity.

The history-guided diffusion model uses a context window of 8 frames. To generate more than 8 consistent keyframes, we employ a two-stage inference scheme. First, we generate 8 keyframes that uniformly span the entire trajectory using only the provided input image as conditioning. Then, we generate the remaining keyframes using the same model, with the nearest already generated keyframes as conditioning. This ensures coherence across arbitrarily long keyframe sequences while keeping the diffusion cost manageable.

3.3. 3D Reconstruction and Rendering

Given the generated keyframes, we reconstruct a 3D representation of the static scene and render the dense video along the trajectory. We use the pretrained AnySplat model [22], which predicts a 3D Gaussian Splatting (3DGS) representation directly from a small set of unposed images.

3.3.1. Preliminaries

3D Gaussian Splatting. 3DGS [23] represents a scene using anisotropic Gaussian primitives parameterized by their mean, covariance, color, and opacity. Rendering is performed via differentiable rasterization in screen space. 3DGS offers real-time rendering performance while maintaining high visual fidelity.

AnySplat Reconstruction. AnySplat predicts Gaussian parameters from multiple unposed images in a single forward pass. Instead of iterative inverse rendering, it maps multi-view features to a 3D Gaussian field, enabling fast and reliable sparse-view reconstruction. It uses VGGT [48] to estimate camera poses, permitting training on unposed datasets. AnySplat is deterministic and reconstructs only the visible scene content. It processes tens of images in a few seconds.

3.3.2. Our Model.

We feed the generated keyframes into AnySplat to obtain a 3D Gaussian representation of the scene. Since AnySplat's predicted poses and the input trajectory lie in different coordinate frames, we align them by estimating a least-squares affine transform Finally, we render the dense output video by evaluating the 3DGS renderer at each camera pose. This stage is extremely fast and accounts for much of `SRENDER`'s efficiency relative to diffusion-based baselines.

3.4. Temporal Chunks

Our method relies on the generated keyframes being geometrically and photometrically consistent across the full camera trajectory. For simpler datasets such as RealEstate10k, and for short durations in more complex scenes, this assumption generally holds: diffusion models produce keyframes stable enough for a single high-quality global reconstruction.

However, on more challenging datasets such as DL3DV, we observe noticeable drift in the generated keyframes for long trajectories, using our model or other baseline models. Keyframes beyond roughly 10 seconds often exhibit inconsistencies in structure, appearance, or relative geometry. This reflects a broader limitation of current diffusion models, which struggle with long-range loop-closure and maintaining multi-view consistency across large-baseline changes. Attempting to reconstruct a single global 3D scene from all keyframes leads to blurred reconstructions, as the 3D model must reconcile mutually inconsistent observations. To address this, we divide the keyframes into fixed-length *temporal chunks*. In practice, we use chunk durations of 10 seconds, within which the generated keyframes are empirically consistent.

For each chunk, we perform an independent 3D reconstruction using AnySplat and align the dense input camera trajectory individually. Since each chunk produces its own 3D Gaussian representation in its own coordinate frame, we include a shared keyframe between adjacent chunks and align the scenes by estimating a affine transformation between the shared pose sets. After alignment, we render the dense video by querying the appropriate chunk-specific 3D model at each camera pose. This chunked reconstruction strategy allows our method to generate high-quality videos

without requiring the diffusion model to generate drift-free keyframes.

4. Experiments

4.1. Experimental Details

Datasets. We conduct experiments on two camera-conditioned video datasets: RealEstate10k (RE10K) [12] and DL3DV [30]. For RE10K, we generate 20-second videos at 10 fps (200 frames) along the provided camera paths. For DL3DV, which contains significantly larger camera motions, we generate 20-second videos at 30 fps (600 frames). We train separate models for each dataset, and train all models at a resolution of 256×256. For evaluation, we use 50 videos from the DL3DV test set and 200 videos from the RE10K test split. Outside of the primary evaluation in Table 1, we use a subsampled 5 fps version of DL3DV, as generating high-frame-rate sequences with the baseline methods is either very slow or not possible. This test set allows us to perform comparisons on long sequences without high computational costs.

Video Generation Baselines. Our primary baseline is the History-Guided Video Diffusion model (HG) [42]. This baseline is particularly relevant because it shares the same diffusion architecture and training setup as our keyframe generator; the only difference is that HG generates every frame, whereas the keyframe generator in SRENDER generates only sparse keyframes. We train HG on the same training splits as our model.

In addition, we compare against the state-of-the-art camera-conditioned model *Voyager* from Hunyuan [19]. Voyager is a very recent model that benchmarks against and improves upon multiple modern video generation approaches [41, 54, 61], making it a strong representative baseline. We use the pretrained model of Voyager. The currently available implementation of Voyager is not capable of generating videos with hundreds of frames; therefore, we only compare it in the 5 fps setting.

2D Interpolation Baselines. To evaluate the importance of 3D reconstruction, we compare against two 2D frame interpolation methods: FILM [39] and RIFE [21]. These methods interpolate intermediate frames between our generated keyframes, allowing us to assess whether 3D rendering offers advantages over purely 2D temporal interpolation.

4.2. Evaluation Metrics

Image and Video Quality. We evaluate per-frame image quality using the Fréchet Inception Distance (FID) [15], which measures the distributional similarity between generated frames and ground-truth frames. To assess temporal coherence, we use the Fréchet Video Distance (FVD) [46], which measures both appearance and motion consistency by comparing distributions of video clips extracted from the generated and ground-truth sequences.

Efficiency. A central goal of SRENDER is efficient video generation. We therefore report the *generation time* required to synthesize each video, measured in wall-clock time on a single NVIDIA GH200 Superchip. The speed numbers reported for the experiments on full video generation include both keyframes generation and the subsequent 3D reconstruction and rendering, and the numbers reported for the interpolation baselines include only interpolation.

4.3. Quantitative Results

Table 1 summarizes the quantitative comparison on RE10K and DL3DV. Across both datasets, SRENDER outperforms the History-Guided Video Diffusion (HG) baseline on both quality metrics, FID and FVD. This demonstrates that replacing dense diffusion with our keyframe diffusion and 3D rendering pipeline does not degrade visual quality, and in fact yields more consistent image and video generation.

The primary advantage of SRENDER is its computational efficiency. On DL3DV, we achieve more than **40**$\times$ speed-up over HG, and more than **20**$\times$ speed-up on RE10K. Notably, SRENDER reaches real-time performance on DL3DV, requiring only **16.21 seconds** on average to generate a 20-second video at 256×256 resolution, corresponding to a generation framerate of **37.01 fps**. In comparison, HG achieves a generation framerate of **0.86 fps**. These results validate that explicit 3D reconstruction and rendering provide substantial efficiency gains without sacrificing fidelity.

Table 2 reports results on the 5 fps test set of DL3DV, where we compare against both the HG and the state-of-the-art Voyager model. Across all metrics, SRENDER again achieves superior image quality (FID) and video quality (FVD), while being significantly faster than both baselines.

4.4. Qualitative Results

Figure 3 shows qualitative comparisons on DL3DV against the HG and the Voyager model. Our method produces high-quality results under large viewpoint changes. Figure 4 provides additional comparisons on RE10K, where SRENDER again achieves high visual fidelity and consistent appearance across the trajectory. We refer readers to the supplemental video for full visualizations of the generated sequences. Although the rendered videos from our 3D reconstruction pipeline may appear slightly smoother and contain fewer high-frequency details than the diffusion baselines, they avoid the characteristic high-frequency artifacts produced by the HG model and maintain significantly better geometric stability than Voyager, exhibiting consistent structure and appearance across the entire trajectory.

Figure 3. **Qualitative comparisons on DL3DV.** All methods generate 20-second videos at 5 fps, conditioned on the input image and target trajectory. Output frames at 4s and 14s are visualized. Our method achieves both high video quality and camera control. HG [42] often has high-frequency artifacts. Voyager [19] fails at generating a consistent long video sequence. We also show results with 2D interpolation methods [21, 39], which show strong morphing effects, and also cannot satisfy the intermediate camera control inputs.

	DL3DV (20s @ 30 fps, 600 frames)				RE10K (20s @ 10 fps, 200 frames)			
Method / Metric	FID ↓	FVD ↓	Time (s) ↓	Speed-up ↑	FID ↓	FVD ↓	Time (s) ↓	Speed-up ↑
HG [42]	66.89	367.5	697.38	1×	39.53	194.0	226.5	1×
Ours	**60.90**	**335.5**	**16.21**	**43.02**×	**30.23**	**180.3**	**9.552**	**23.71**×

Table 1. **Quantitative comparison on DL3DV and RE10K**. We achieve better results quantitatively at significantly higher speeds.

Method	FID ↓	FVD ↓	Time (s) ↓	Speed-up ↑
Voyager [19]	91.75	808.0	332.0	1×
HG [42]	62.78	497.8	116.0	2.86×
Ours	**61.18**	**492.8**	**13.62**	**24.38**×

Table 2. **Quantitative comparisons on DL3DV (5fps).** We achieve better or comparable quality, with significant speed-ups.

Method	FID ↓	FVD ↓	Time (s) ↓	Speed-up ↑
FILM [39]	**58.94**	619.0	315.0	1×
RIFE [21]	59.72	653.0	2.67	117.98×
Ours	65.87	**482.0**	**0.83**	**379.52**×

Table 3. **Comparison with 2D Interpolation Methods on DL3DV.** All methods are given the same sets of sparse keyframes ∼3s apart, generated by our keyframe generation model. The baselines have significant morphing effects, as reflected in the high FVD scores. In addition, both baselines are significantly slower. Only interpolation time is reported here.

4.5. Ablation Studies

3D vs. 2D Interpolation. A natural question is whether the dense video can be generated by applying a 2D interpolation method to the keyframes, rather than performing 3D reconstruction and rendering. Figure 3 (rightmost two columns) and Table 3 compare our approach with two state-of-the-art 2D interpolation models, FILM [39] and RIFE [21]. Our method outperforms both baselines in terms of FVD and avoids the morphing and warping artifacts commonly observed when interpolating across large viewpoint changes. Interestingly, `SRENDER` is also *faster* than the 2D interpolation baselines, since 3DGS rendering is highly efficient and scales well to long sequences. These results highlight the importance of explicit 3D reasoning for generating geometrically consistent videos.

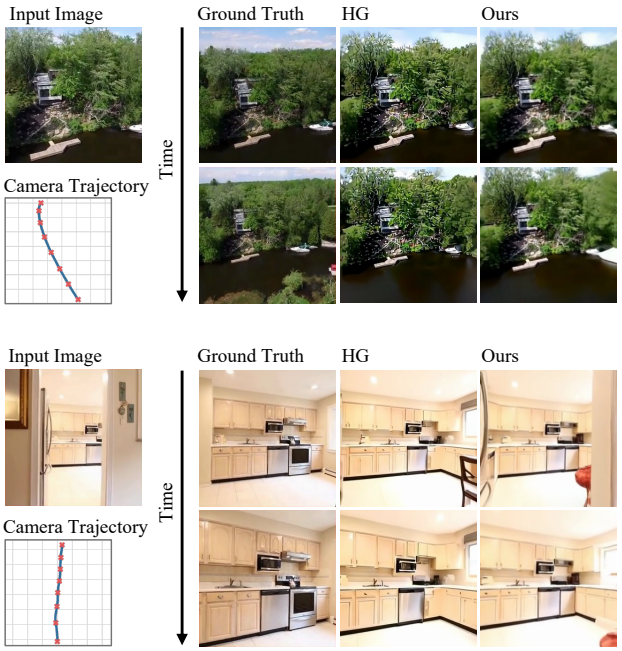

Figure 4. **Qualitative comparisons on RE10K.** Compared with HG [42], our method does not have high-frequency artifacts and is significantly faster.

Method	FID ↓	FVD ↓	Time (s) ↓
HG [42]	63.00	346.5	491.0
Ours (without chunking)	62.84	357.5	13.52
Ours (with chunking)	**59.19**	**336.5**	**13.24**

Table 4. **Evaluation of temporal chunking on DL3DV.** Both FID and FVD are improved with temporal chunking, while the computational time used is comparable.

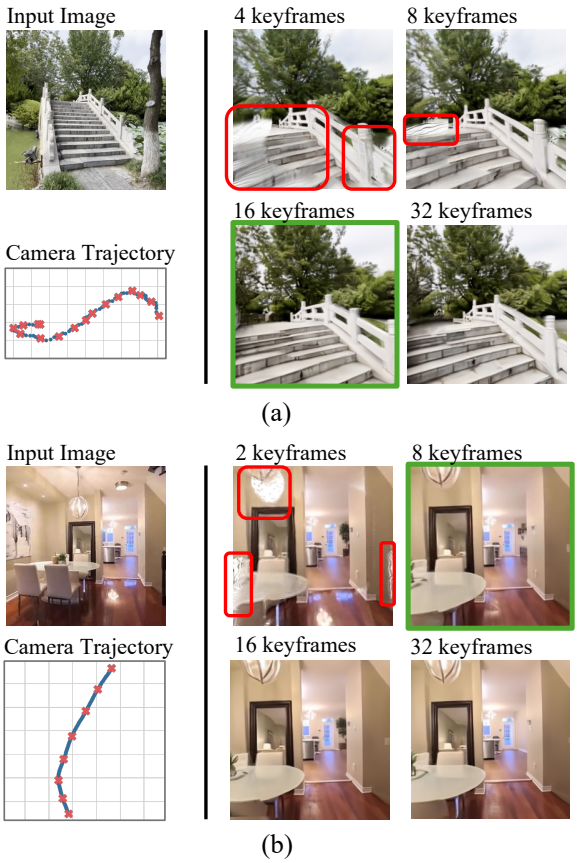

Figure 5. Too few keyframes lead to visible holes in the generated video (red boxes), while generating too many keyframes is significantly more expensive, without significant quality gains. SRENDER selects the optimal number of keyframes (green) that strikes a good balance between completeness and efficiency.

Effect of Temporal Chunking. We evaluate the quantitative performance of our method when generating long, high-frame-rate videos from the DL3DV test dataset, with or without dividing the keyframes into chunks and constructing separate 3D Gaussians. As shown in Table 4, both FID and FVD are improved when temporal chunking is used. Under the temporal chunking setting, the 3D reconstruction model can produce 3D scenes with higher consistency and reduced blurriness, and this is essential for rendering high-quality videos from the scene. Running the 3D reconstruction model multiple times also does not slow down the overall process. The model needs to produce the 3D scene from fewer keyframes each time, making each run faster than a single run over all keyframes together.

Keyframe Selection. A key design choice of our method is that we train a model to predict the keyframe density given an input image and a camera trajectory. In Figure 5, we can see that when too few keyframes are selected, the 3D scene is underdefined, and there will be multiple visible blank areas in the rendered video. However, when enough keyframes can define the scene along the given trajectory, adding more keyframes does not necessarily improve the quality and will add to computational redundancy. Our keyframe model chooses the optimal keyframe density that balances final video quality and computational cost.

5. Discussion

Static Scenes. Our method currently only applies to static scenes. We emphasize, however, that camera-conditioned generation of static videos is an important and well-studied problem in its own right, with numerous recent works focusing exclusively on this setting [14, 19, 26]. Moreover, 4D reconstruction of dynamic scenes is progressing rapidly [27, 49, 50, 59], and as these models mature, the core ideas behind SRENDER, i.e., sparse view generation,

adaptive keyframing, and 3D rendering, will directly be transferable to dynamic environments. We view our method as establishing a foundation for efficient generation that can be extended to full dynamic scenes in future work.

High-frequency Details. Our 3D-rendered videos may appear slightly smoother or less detailed at high frequencies compared to purely diffusion-based baselines. At the same time, they avoid the characteristic artifacts of diffusion models, such as noise amplification or view-dependent distortions, and maintain strong geometric consistency across challenging camera trajectories. As demonstrated in our experiments, this trade-off leads to quantitatively better FID/FVD scores and substantially faster generation. We also note that many practical applications, such as in embodied AI, do not require the highest-frequency details, but instead prioritize global coherence and structural stability. Finally, as 3D reconstruction models improve, we expect the visual fidelity of our results to increase correspondingly.

6. Conclusion

We presented a simple yet effective approach for efficient camera-conditioned video generation. By explicitly leveraging the inherent redundancy in video data, our method produces long, geometrically consistent videos at a small fraction of the computational cost of existing diffusion-based models. Our experiments demonstrate that this strategy not only yields dramatic speed-ups, but also maintains or improves visual quality relative to strong baselines. We believe that the core ideas behind `SRENDER` will serve as a foundation for future work on efficient video synthesis.

References

[1] Sand. ai, Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, W. Q. Zhang, Weifeng Luo, Xiaoyang Kang, Yuchen Sun, Yue Cao, Yunpeng Huang, Yutong Lin, Yuxin Fang, Zewei Tao, Zheng Zhang, Zhongshu Wang, Zixun Liu, Dai Shi, Guoli Su, Hanwen Sun, Hong Pan, Jie Wang, Jiexin Sheng, Min Cui, Min Hu, Ming Yan, Shucheng Yin, Siran Zhang, Tingting Liu, Xianping Yin, Xiaoyu Yang, Xin Song, Xuan Hu, Yankai Zhang, and Yuqiao Li. Magi-1: Autoregressive video generation at scale, 2025. 2

[2] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets, 2023. 2

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

[4] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024. 4, 5

[5] Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion, 2024. 2

[6] Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, Weiming Xiong, Wei Wang, Nuo Pang, Kang Kang, Zhiheng Xu, Yuzhe Jin, Yupeng Liang, Yubing Song, Peng Zhao, Boyuan Xu, Di Qiu, Debang Li, Zhengcong Fei, Yang Li, and Yahui Zhou. Skyreels-v2: Infinite-length film generative model, 2025. 2

[7] Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-forcing++: Towards minute-scale high-quality video generation, 2025.

[8] Decart. Miragelsd.

[9] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization, 2025. 2

[10] NVIDIA et. al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 2

[11] Ruiqi Gao*, Aleksander Holynski*, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul P. Srinivasan, Jonathan T. Barron, and Ben Poole*. Cat3d: Create anything in 3d with multi-view diffusion models. *Advances in Neural Information Processing Systems*, 2024. 3

[12] Google. Realestate10k: A large dataset of camera trajectories from video clips. 2, 4, 6

[13] Google Deepmind. Veo3 — google deepmind. 2

[14] 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. 2, 8

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. 6

[16] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 5

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 2, 4

[18] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022. 2

[19] Tianyu Huang, Wangguandong Zheng, Tengfei Wang, Yuhao Liu, Zhenwei Wang, Junta Wu, Jie Jiang, Hui Li, Rynson W. H. Lau, Wangmeng Zuo, and Chunchao Guo. Voyager: Long-range and world-consistent video diffusion for explorable 3d scene generation, 2025. 2, 3, 6, 7, 8, 13

[20] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion, 2025. 2

[21] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 6, 7, 13

[22] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views, 2025. 2, 3, 5, 12

[23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023. 2, 3, 5

[24] World Labs. Marble: A multimodal world model. 3

[25] Jiachen Li, Weixi Feng, Tsu-Jui Fu, Xinyi Wang, Sugato Basu, Wenhu Chen, and William Yang Wang. T2v-turbo: Breaking the quality bottleneck of video consistency model with mixed reward feedback, 2024. 3

[26] Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. *Advances in Neural Information Processing Systems*, 2025. 8

[27] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. Megasam: Accurate, fast, and robust structure and motion from casual dynamic videos, 2024. 8

[28] Shanchuan Lin and Xiao Yang. Animatediff-lightning: Cross-model diffusion distillation, 2024. 3

[29] Wenfeng Lin, Renjie Chen, Boyuan Liu, Shiyue Yan, Ruoyu Feng, Jiangchuan Wei, Yichen Zhang, Yimeng Zhou, Chao Feng, Jiao Ran, Qi Wu, Zuotao Liu, and Mingyu Guo. Contentv: Efficient training of video generation models with limited compute, 2025. 2

[30] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision, 2023. 2, 4, 6

[31] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 1

[32] Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time, 2025. 2

[33] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 3

[34] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023. 3

[35] Yifan Liu, Zhiyuan Min, Zhenwei Wang, Junta Wu, Tengfei Wang, Yixuan Yuan, Yawei Luo, and Chunchao Guo. Worldmirror: Universal 3d world reconstruction with any-prior prompting. *arXiv preprint arXiv:2510.10726*, 2025. 2, 3

[36] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9970–9980, 2024. 3

[37] OpenAI. Sora — openai. 1, 2

[38] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. 4

[39] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *European Conference on Computer Vision (ECCV)*, 2022. 6, 7, 13

[40] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control, 2025. 2

[41] Runway Research. Runway research — introducing gen-3 alpha: A new frontier for video generation. 6

[42] Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion, 2025. 1, 2, 4, 5, 6, 7, 8, 12, 13

[43] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Viewset diffusion:(0-) image-conditioned 3d generative models from 2d data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8863–8873, 2023. 3

[44] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation, 2024. 3

[45] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Joshua B. Tenenbaum, Frédo Durand, William T. Freeman, and Vincent Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. In *arXiv*, 2023. 3

[46] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges, 2019. 6

[47] Team Wan, 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. *arXiv preprint arXiv:2503.20314*, 2025. 1, 2

[48] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 3, 4, 5, 12

[49] Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video, 2025. 8

[50] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025. 8

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

[52] Xiang Wang, Shiwei Zhang, Han Zhang, Yu Liu, Yingya Zhang, Changxin Gao, and Nong Sang. Videolcm: Video latent consistency model, 2023. 3

[53] Haoyu Wu, Diankun Wu, Tianyu He, Junliang Guo, Yang Ye, Yueqi Duan, and Jiang Bian. Geometry forcing: Marrying video diffusion and 3d representation for consistent world modeling, 2025. 2

[54] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2025. 2, 6

[55] Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. 2025. 2, 3

[56] Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T. Freeman, and Jiajun Wu. Wonderworld: Interactive 3d scene generation from a single image. *arXiv:2406.09394*, 2024. 3

[57] Sihyun Yu, Weili Nie, De-An Huang, Boyi Li, Jinwoo Shin, and Anima Anandkumar. Efficient video diffusion models via content-frame motion-latent decomposition, 2024. 2

[58] Hangjie Yuan, Weihua Chen, Jun Cen, Hu Yu, Jingyun Liang, Shuning Chang, Zhihui Lin, Tao Feng, Pengwei Liu, Jiazheng Xing, Hao Luo, Jiasheng Tang, Fan Wang, and Yi Yang. Lumos-1: On autoregressive video generation from a unified model perspective, 2025. 2

[59] 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. *arXiv preprint arxiv:2410.03825*, 2024. 8

[60] Jensen Zhou, Hang Gao, Vikram Voleti, Aaryaman Vasishta, Chun-Han Yao, Mark Boss, Philip Torr, Christian Rupprecht, and Varun Jampani. Stable virtual camera: Generative view synthesis with diffusion models, 2025. 2

[61] Yuan Zhou, Qiuyue Wang, Yuxuan Cai, and Huan Yang. Allegro: Open the black box of commercial-level video generation model, 2024. 6

[62] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching, 2025. 3

A. Further Method Details

A.1. Adaptive Keyframe Selection Model

A.1.1. Detailed Model Architecture

The model takes a dense camera trajectory and a reference image as input and outputs the required number of sparse keyframes. It comprises four components: a camera embedding module, a DINOv2 encoder, a transformer, and a final output MLP.

We embed the camera trajectory by converting the 3×3 rotation matrices into 4D quaternions, appending them to the translation vectors to form 7D vectors, and projecting them to have the same dimensionality as DINOv2 using a 2-layer MLP. From the DINOv2 encoder, we process the reference image and retain only the global class token, discarding all other image tokens. This token is appended to the sequence of camera tokens. The transformer, configured with 4 heads and 4 layers, processes this sequence and produces output tokens. The output tokens are passed through a 4-layer MLP to regress the final number of sparse keyframes.

A.1.2. Training

To stabilize the training, instead of regressing a single number as the keyframe density, we have the final MLP layers regress one number for each of the output tokens from the transformer and take the average of them as the final output of the keyframe selection model. Additionally, as the keyframe counts in our curated dataset are in the range of 4 to 35, we find that scaling the final output by 0.1 regulates the output and improves the stability of the training. Specifically, the training loss of this model can be written as

$$\mathcal{L} = 0.1 * \mathbb{E}\big[\,(\bar{y} - n_{gt})^2\big] \tag{1}$$

where $\bar{y}$ is the average of the output vector of the MLP, and n_{gt} is the ground truth keyframe number. We optimize the network over the entire dataset.

We train the model using AdamW with a learning rate of 1e-4 and a batch size of 128. Training was performed on a single NVIDIA GH200 Superchip for 5 hours.

A.1.3. Uniform Keyframe Sampling

While directly predicting the exact keyframe indices would be ideal, this objective is highly unstable to train due to the inherent ambiguity of the problem. Instead, we regress the total number of sparse keyframes required and uniformly

distribute them along the camera trajectory, which yields better results.

A.2. Keyframe Generation Model

Our keyframe generation model is a history-guided video diffusion model (HG model) [42]. We train a camera-controlled, diffusion-forcing transformer (DFoT) on a sparse keyframe dataset with camera pose annotations.

A.2.1. Training

Conventional video diffusion models are trained to denoise an entire video from the same noise level. In contrast, the diffusion-forcing architecture assigns an independent noise level to each frame and trains the model to denoise all the frames from the different noise levels together, minimizing the noise prediction loss

$$\mathcal{L} = \mathbb{E}\left[\left\|\epsilon_{\mathcal{T}} - \epsilon_\theta\left(x_{\mathcal{T}}^{k_{\mathcal{T}}},\, k_{\mathcal{T}}\right)\right\|^2\right], \tag{2}$$

where τ is the set of frame indices in a video, $x_{\mathcal{T}}^{k_{\mathcal{T}}}$ denotes noisy frames, $k_{\mathcal{T}}$ denotes the associated noise levels, and $\epsilon_{\mathcal{T}}$ is the noise added to the frames. $\epsilon_\theta()$ is a neural network that predicts the added noise, and θ denotes the model parameters to be optimized. The parameters are optimized to minimize the noise prediction loss across all frames and noise levels.

Diffusion-forcing models trained in this manner can use either input frames or their own generated outputs as guidance, enabling autoregressive sampling and infinite video generation. They can also accept guidance at arbitrary positions, enabling interpolation. For camera control, we follow HG and encode camera poses into ray maps for conditioning.

When training our keyframe generation model, we start from the RE10K checkpoint provided by [42] and finetune for the DL3DV dataset. As stated in Section 3.2.2, we first train the model with consecutive frames and then progressively increase the gap between adjacent frames to approximately 4 seconds, matching the keyframe density predicted by our keyframe selection model. We trained the model with a batch size of 8 and a learning rate of 5e-5 on 8 NVIDIA GH200 Superchips for 6 days.

A.2.2. Inference

DFoT is trained to denoise frames with independent noise levels. At inference time, the image conditioning (or 'history') is applied by giving conditional frames to the model with zero noise level.

For all our experiments, we use the vanilla history guidance and adhere to the default parameter settings as stated in the HG GitHub repository[1]. Our model uses a context window of 8 frames. When generating more than 8 keyframes, we first generate 8 keyframes that span the entire trajectory using only the provided input image as conditioning. Then we generate the remaining keyframes using the nearest already generated keyframes as conditioning.

[1] https://github.com/kwsong0113/diffusion-forcing-transformer

A.3. 3D Reconstruction Model

As specified in Section 3.3, we use AnySplat [22] as our 3D reconstruction model to obtain the 3D Gaussian scene representation from the generated keyframes. AnySplat takes uncalibrated images as input and uses a transformer-based geometry encoder to extract the latent representations of the images. It then utilizes three decoder heads to predict the Gaussian parameters of the scene, the depth map, and the camera poses associated with each input image. Finally, the Gaussian parameters are voxelized into per-voxel 3D Gaussians using a Differentiable Voxelization module. The model is trained using an RGB loss between the input images and the renderings from the predicted 3D Gaussians, along with a geometry loss that uses the corresponding estimates from the pretrained VGGT model [48] to supervise the predicted camera poses and depth maps. The RGB loss consists of a mean-squared error term and a Learned Perceptual Image Patch Similarity (LPIPS)-based loss. The geometry loss is the mean-squared error between the depth maps and camera encoding estimated by the two models. In our pipeline, we feed AnySplat our generated keyframes and obtain the 3D Gaussians and the predicted keyframe camera poses for subsequent processing.

A.4. Camera Pose Alignment for Rendering

A.4.1. Aligning the Camera Trajectory with the 3D Reconstruction

The input camera trajectory and 3D Gaussians predicted by AnySplat may differ in scale, rotation, and translation. We compute a similarity transformation to align the input trajectory with the 3D reconstruction's coordinate system. We do so by first estimating the least squares similarity transformation between AnySplat's predicted camera poses for the keyframes and the camera poses for the keyframes from the full input camera trajectory.

$$S^* = \arg\min_{S \in \mathrm{Sim}(3)} \sum_{k \in K} \left\| S \circ \Phi_k^{\mathrm{in}} - \Phi_k^{\mathrm{as}} \right\|^2, \tag{3}$$

Then, we apply the estimated transformation to the whole input camera trajectory.

$$\widehat{\Phi}_i^{\mathrm{as}} = S^* \circ \Phi_i^{\mathrm{in}}. \tag{4}$$

Here, K is the set of keyframes, Φ_k^{in} denotes the keyframe camera poses in the full input trajectory, and Φ_k^{as} denotes the keyframe camera poses predicted by AnySplat for the 3D Gaussians. $S = (s, R, t) \in Sim(3)$ is the similarity transformation, and $S \circ \Phi$ denotes applying the similarity transformation S to the camera pose Φ. Finally, we

render the whole video using the transformed full camera trajectory $\widehat{\Phi}_i^{\text{as}}$.

A.4.2. Aligning the Camera Trajectory under Temporal Chunking

Similarly, when we divide the video into temporal chunks and reconstruct each chunk independently, each chunk may have its own coordinate system. To produce a seamless video, we include a shared keyframe between consecutive chunks and align the dense input camera trajectory to each chunk individually. We further compute and apply a rigid body transformation to the initially aligned camera trajectory to ensure the transformed shared keyframe poses at each chunk are identical to the shared keyframe poses predicted by AnySplat. This step is essential for a smooth transition across the chunk boundary. The full video is rendered chunk by chunk, and duplicate renders of the overlapping keyframes are discarded.

B. Further Experimental Details

B.1. Datasets

The RE10K dataset used in our experiments was provided by History-Guided Video Diffusion [42] and was obtained from https://huggingface.co/kiwhansong/DFoT/tree/main/datasets . All videos are at 256×256 image resolution. We evaluate on 200 frames at 10 fps.

The DL3DV evaluation set is downloaded from https://huggingface.co/datasets/DL3DV/DL3DV-Evaluation. We resize all videos to a height of 256 pixels and center-crop them to obtain square frames. The dataset provides camera poses at 5 fps; we interpolate them using slerp to obtain camera poses at 30 fps. We evaluate on 100 frames at 5 fps. We also evaluate DL3DV on a high frame rate setting of either 400 or 600 frames at 30 fps.

B.2. Detailed Experiment Setup

Following the default setting of HG [42] for generating 200-frame 10 fps videos from the RE10K dataset, we compute the baseline results of the HG model on the RE10K dataset by first generating 12 frames uniformly spanning the full trajectory (keyframes) with the input image as conditioning, and then generating the remaining frames using the nearest two keyframes as conditioning. We maintain the same initial keyframe density in time for the HG baseline for all experimental setups, which is 12 keyframes in 20 seconds. The original HG model was not trained on the DL3DV dataset. For a fair comparison, we train a model with a 2-second jump on the DL3DV dataset as the baseline model. This is comparable to the training setting of the HG model for the RE10K dataset in the original paper [42].

B.3. Further Experiments at High Frame Rate

Under the high frame rate setting in DL3DV, we also generate 600 frames at 30 fps or 20 seconds of video. Quantitatively, our method achieves better FID and FVD with a >40x speed-up, as shown in Table 5 of the supplemental. As our method generates a 20-second video using only 16.21 seconds on average, we generate videos faster than real-time. Qualitatively, the HG model produces more prominent flickering and shaky camera artifacts when generating longer videos. Our method does not have these drawbacks. More visual results can be found on our supplemental webpage.

Method	FID ↓	FVD ↓	Time (s) ↓	Speed-up ↑
HG [42]	66.89	367.5	697.38	1×
Ours	**60.90**	**335.5**	**16.21**	**43.02**×

Table 5. **Quantitative comparisons on DL3DV (20s@30fps).** We achieve better or comparable quality, with even more significant speed-ups.

B.4. Comparisons with Voyager

Since Voyager [19] was trained on rectangular images, but we evaluate on square images, for the DL3DV dataset, we use the full image for generation and crop to square for evaluation. This gives slightly higher performance than performing inference with the square image directly.

In their paper, Voyager claims to be able to generate arbitrarily long videos with autoregressive sampling and World Caching with Point Culling. However, this part of the method is not found in their official implementation[2]. Thus, we only compare in the 100 frame at 5 fps setting. We show qualitative results on our website, and quantitative results in Table 2 of the main paper.

B.5. Comparisons with 2D interpolation methods

We compare our interpolation method using 3D reconstruction with two 2D video interpolation methods, RIFE [21] and FILM [39]. We first generate keyframes using our keyframe generation model at a fixed keyframe density. Then, we use the same set of keyframes for the 2D interpolation methods and our method to interpolate the full video.

Our method shows superior results qualitatively and quantitatively, as shown in Table 3 (main paper). Both 2D methods do not consider camera control when interpolating between frames, resulting in naively morphing when the camera control is complex.

[2]https://github.com/Tencent-Hunyuan/HunyuanWorld-Voyager

C. Videos of the Same Scene with Different Camera Trajectories

After generating the 3D Gaussian and a video with our method, we can render new videos with different camera trajectories, even those that are out-of-distribution with respect to the trajectories observed during training, in seconds. The baseline diffusion model, on the other hand, would need to rerun the generation process, which requires hundreds of seconds. Please refer to the supplemental webpage for the qualitative results.