SierpinskiCam: Camera-Controlled Video Retaking with Sierpinski Triangle Pattern Cues

Suttisak Wizadwongsa[1,2]* **Hyelin Nam**[1]* **Supasorn Suwajanakorn**[2] **Jeong Joon Park**[1]†
[1]University of Michigan, Ann Arbor [2]VISTEC, Thailand

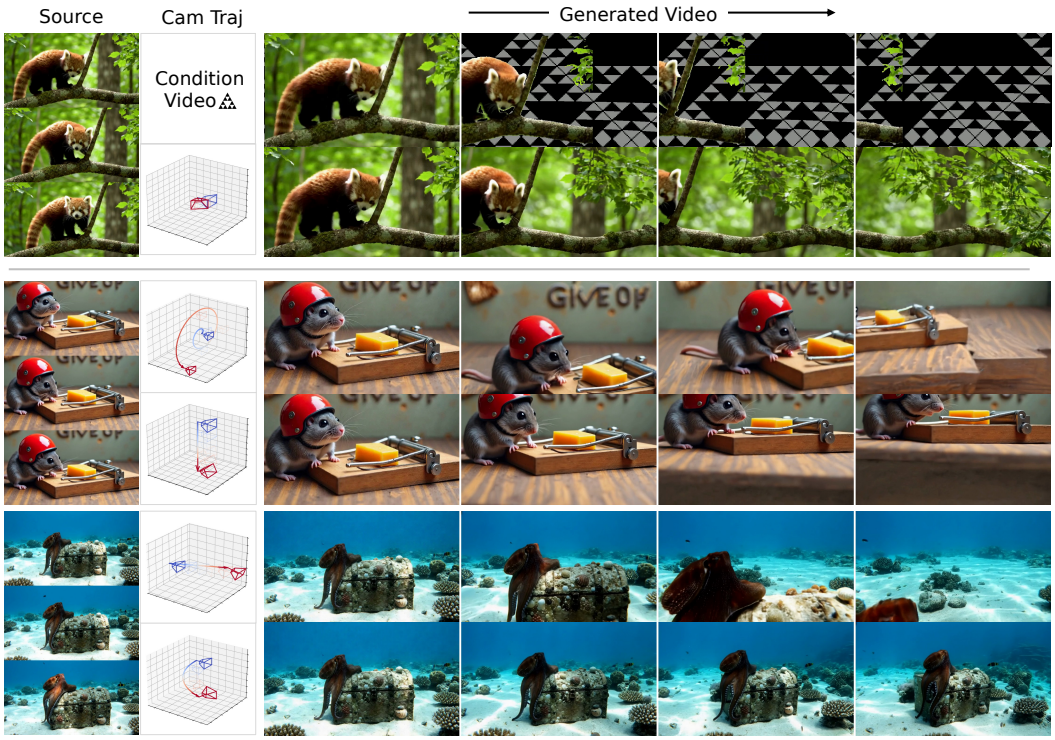

Figure 1: **SierpinskiCam for video retake generation.** Given a source video and a target camera trajectory (blue → red), SierpinskiCam retakes the video under user-defined camera motions. Even under large viewpoint changes with sparse source evidence, our Sierpinski textured dome (top *condition video*) and negative rotary position embedding allow faithful following of the target camera trajectory while preserving the original scene content. Source videos are generated by Veo3 [7].

Abstract

Generating novel renderings of a scene along user-defined camera trajectories from a single monocular video, dubbed video retaking, is a compelling but difficult problem in content creation and visual effects. Existing geometry-guided approaches reconstruct a 4D representation from the source video and render it along the target trajectory to condition video diffusion models. However, this guidance degrades as the target camera departs from the source trajectory, leaving newly revealed regions sparse or entirely missing. We propose SierpinskiCam, which addresses this limitation by augmenting geometry-based guidance with Sierpinski dome texture cues that contain rich trackable features even under large viewpoint changes. We further

*Equal contribution.
†Corresponding author.

Preprint.

introduce a reference video conditioning mechanism that appends source-video tokens to the target-token sequence and separates the two streams with negative RoPE indices, enabling appearance grounding without architectural modification or per-video adaptation. Extensive experiments show that SierpinskiCam achieves significant gains in camera controllability, geometric consistency, and video quality across diverse and challenging retaking scenarios.

1 Introduction

Video retaking aims to re-render an existing video under a new camera trajectory while preserving the scene identity, object appearance, and temporal dynamics of the original capture, enabling new creative possibilities in filmmaking, virtual production, and user-generated content. The main challenge is to follow the target camera motion without losing source fidelity, even when the target view reveals regions that were not visible in the original video.

Recent advances in video diffusion models (VDMs) [10, 23, 28] provide a strong generative prior for realistic appearance, temporal coherence, and plausible completion of unseen regions. Adapting these models for video retaking, however, requires addressing two conditioning questions: (i) how to represent and inject the target camera trajectory for precise viewpoint control, and (ii) how to inject the source video for faithful preservation of the original content.

For target camera control, existing methods fall into implicit and explicit formulations. Implicit methods [2, 17, 22] encode the target camera trajectory as pose representations, such as 6-DoF parameters, Plücker rays, or learned embeddings, and inject them into the generative backbone. The model must then infer from data how to disentangle camera motion from scene dynamics and how to interpret the trajectory scale relative to each source video, leaving camera control ambiguous.

Explicit methods [4, 9, 18, 27, 29, 30] reduce this ambiguity by grounding camera control in geometric transformations of the source video. They estimate depth, point tracks, or point clouds from the source video, warp it under the target camera trajectory, and pass the result to a VDM to refine and synthesize missing regions. This provides an interpretable signal that directly specifies where observed content should appear in the target view. However, geometry derived from a monocular source video provides only partial coverage: when the target camera reveals regions outside the original observation, the warped proxy becomes sparse, with few valid scene points and large empty regions devoid of meaningful motion cues. Such uninformative guidance increases the risk that the generative model fails to follow the target trajectory or hallucinates implausible content.

In addition to target-camera guidance, video retaking must condition on the source video to preserve content appearance and dynamics. This is non-trivial because the source video is not spatially aligned with the target view, making direct feature reuse unreliable. Existing methods address this through channel-wise inputs [22], dedicated reference-attention modules [29], or per-video fine-tuning [4, 30]. However, these designs require architectural modifications or video-specific adaptation that complicate deployment and limit generalization across different backbones, resolutions, and sequence lengths.

To address these limitations, we introduce SierpinskiCam, which tackles both conditioning questions with a Sierpinski-patterned texture dome and position-disentangled source injection. For target camera control, we complement geometry-based proxy rendering with a textured dome that fills empty target-view regions with a Sierpinski fractal pattern, a self-similar texture whose multi-scale edges and corners remain trackable across camera distances, turning otherwise uninformative regions into coherent camera-motion cues. For source preservation, we append source-video tokens to the target-token sequence without architectural modification or per-video adaptation, and assign them negative spatial RoPE indices (NegRoPE) to prevent the model from spuriously attending to source and target tokens that share the same spatial position indices rather than semantically related content. In summary, our contributions are: (i) a Sierpinski texture dome that provides dense, scale-robust camera-motion cues in newly revealed regions, (ii) NegRoPE, a position-disentangled source injection mechanism that requires no architectural modification or per-video adaptation, and (iii) comprehensive experiments validating our design choices and demonstrating state-of-the-art performance in camera controllability, geometric consistency, and video quality across diverse and challenging retaking scenarios (Fig. 1, Fig. 4, Tab. 1, and Tab. 3), with 15% higher user preference scores (Tab. 2-(c)).

2 Related work

2.1 Camera Control for Video Retaking

Building on recent advances in video diffusion models (VDMs) [10, 23, 28], recent studies have explored controlling video generation through camera pose conditioning. We organize relevant works into two categories: implicit methods, in which the model must infer geometry on its own, and explicit methods, which rely on external 3D information as constraints during synthesis.

Implicit Methods. Implicit methods [2, 17, 22] inject both sources of control directly into video generative models by conditioning on camera extrinsic parameters and input latents, thereby learning the retaking process through dataset supervision. Recent approaches following this paradigm construct large-scale synthetic triplet datasets of input videos, target camera trajectories, and corresponding retaken videos using rendering pipelines, and train large diffusion models on these datasets. However, because camera poses can vary substantially in distribution and scale across datasets, the model must implicitly infer how the target trajectory relates to the source video from supervision alone. As a result, their performance remains strongly tied to the coverage of synthetic training data, often leading to limited generalization capability. ReDirector [17] mitigates this issue by additionally providing the source camera trajectory and explicitly modeling the relative pose difference through positional encoding; however, it still struggles to fully disentangle camera motion from object motion within the video, reflecting an inherent limitation of implicit methods.

Explicit Methods. On the other hand, explicit approaches [4, 9, 18, 27, 29] approximate the underlying scene geometry to reuse the original video content. They typically estimate depth for each frame, lift the sequence into a 3D proxy, and warp the frames along a new camera trajectory. Several methods combine this procedure with fine-tuning of generative video models to inpaint or refine regions that become invalid after warping. While these geometric cues are effective for reusing observed content, they become less informative when the target camera moves far from the original viewpoint or departs substantially from the observed scene. In such cases, unlike implicit methods that can continue to receive camera pose conditioning, explicit warping provides little additional signal about the intended camera motion. As a result, the generative model may hallucinate content that is inconsistent with the original scene context. We target this failure mode by replacing the commonly used black background with texture patterns that allow the model to continue inferring camera motion even in regions where warped content is unavailable. Recent work [3, 13] addresses the same broad limitation from a different angle by improving the 4D proxy itself. This direction is complementary to ours, as our method provides additional texture-based camera-motion cues that can be applied on top of explicit retaking pipelines.

2.2 Source Preservation in Video Retaking

Preserving the source video is a separate challenge from specifying the target camera motion. Since the source and target views are generally not spatially aligned, prior work differs in how the source video is exposed to the generator. Generative Camera Dolly [22] uses channel-wise latent conditioning, while ReCamMaster [2] concatenates source and target clips along the temporal axis so that 3D self-attention can learn source-target relationships. TrajectoryCrafter [29] instead introduces a dedicated Ref-DiT cross-attention module for reference-video conditioning. Other approaches instead rely on per-video adaptation of the generative prior to preserve source-specific appearance and motion [4, 9, 30]. These strategies provide effective source preservation, but they either introduce additional conditioning modules, rely on per-video adaptation, or tie source conditioning to a fixed interface. Our method also uses token concatenation, but makes it architecture-preserving and position-aware by assigning negative RoPE indices to appended source tokens.

3 Background: Rotary Position Embedding (RoPE)

With the shift from convolutional architectures to transformer-based diffusion models, positional encoding becomes a key component for modeling relationships among latent tokens. RoPE [21] is commonly used as a positional encoding in diffusion transformer (DiT) models, providing a simple and general mechanism for preserving relative positional structure among latent tokens. In

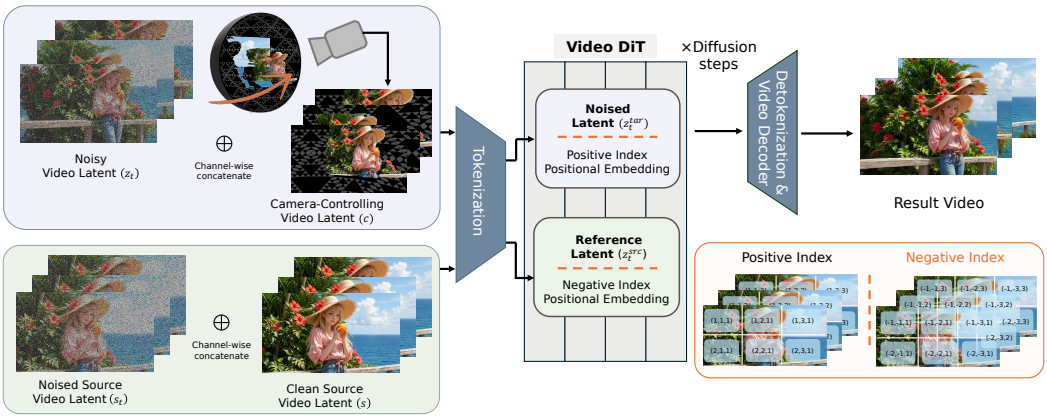

Figure 2: **Overview of SierpinskiCam**. Our model processes two parallel streams: (1) a target stream (top) where noisy target latents z_t are concatenated with the Sierpinski-dome camera-controlling video c, using positive RoPE indices for spatial alignment; and (2) a source stream (bottom) where noised and clean source latents are concatenated using negative RoPE indices. This negative indexing isolates the source content spatially while enabling semantic attention. Both streams are tokenized and processed jointly by the Video DiT to denoise and generate the final result.

DiT-based video diffusion models such as Wan [23], RoPE is applied to patchified latent tokens before computing attention. Specifically, after VAE encoding and patchification, each attention head applies linear projections to obtain queries and keys $q, k \in \mathbb{C}^{N\times(d_{\text{head}}/2)}$, where real and imaginary components represent adjacent channels and d_{head} is even. For a token at position n and channel c, the rotation is applied via

$$\bar{q}_{n,c} = q_{n,c}\exp(i\omega_c n), \qquad \bar{k}_{n,c} = k_{n,c}\exp(i\omega_c n), \tag{1}$$

where the angular frequencies follow an exponential schedule $\omega_c = 10000^{-2(c-1)/d_{\text{head}}}$ for $c = 1, \ldots, d_{\text{head}}/2$. This yields attention scores

$$A'(n,m) = \text{Re}[\bar{q}_n \bar{k}_m^*] = \text{Re}\left[q_n \left(k_m^* \circ \exp(i\omega_c(n-m))\right)\right], \tag{2}$$

which depend only on the relative position difference $(n-m)$, enabling the model to capture spatial relationships independently of absolute token positions. In video diffusion models, this formulation extends to latent tokens by assigning RoPE components along the frame, height, and width axes. In this work, we manipulate only the height and width components, leaving temporal positional encoding unchanged; The details are presented in Sec. 4.2.

4 Methods

In this section, we present SierpinskiCam, a camera-controlled video retaking method that effectively handles target views even when visual guidance from the source video becomes sparse. In Sec. 4.1, we describe how Sierpinski triangle patterns are rendered under the target trajectory to produce dense, persistent camera-motion cues. Then, in Sec. 4.2, we introduce negative rotary position embedding (NegRoPE) for position-disentangled source injection, which appends source-video tokens to the target-token sequence while assigning them negative spatial RoPE indices to avoid direct positional collisions with target tokens.

4.1 Generating Camera-Motion Cues Using Sierpinski Triangle Fractal Patterns

Geometry-based camera-motion proxy. We first construct geometry-based proxies by lifting the source video into 3D and rendering it under the target camera trajectory. Following explicit camera-control pipelines such as TrajectoryCrafter [29], we use dense 3D point clouds reconstructed from monocular depth and sparse 3D point tracks. For the dense proxy, we use DepthAnything-V3 [12] to estimate per-frame depth, extrinsics, and intrinsics, and temporally smooth the estimated source cameras to reduce frame-wise reconstruction jitter. The source pixels are then forward-splatted from the estimated source cameras to the target camera trajectory, yielding a target-view RGB proxy and

a validity mask. In parallel, we obtain sparse 3D point tracks using SpaTracker-V2 [26]; projected tracks are rendered as fixed-radius colored points, with colors sampled from their corresponding first-frame pixels. Although these proxies provide direct geometric guidance where valid source points are available, they often struggle to disentangle global perspective changes from local subject movements, especially when the motion signals are sparse or absent.

Sierpinski textured dome. This motivates an additional camera-motion cue that remains informative beyond the valid coverage of the source-derived geometry. We texture the dome with a Sierpinski fractal triangle pattern, a structured high-contrast pattern that supplies repeated edges and corners across the image. This choice follows the common use of Sierpinski patterns in motion tracking, where reliable features from multi-scale fractal pattern help recover geometric transformations across camera distances [25, 31] (Fig. 3). We analyze this choice in Sec. 5.3.

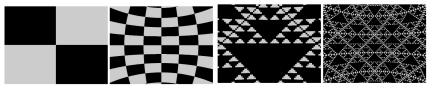

Figure 3: **Multi-scale robustness.** Unlike the Checkerboard pattern (a), the Sierpinski fractal pattern (b) provides structural details in both near and far views, ensuring camera pose control across scales.

We implement this auxiliary cue by rendering a textured spherical dome around the estimated proxy scene geometry. For each target camera, we cast a ray from each pixel using the target intrinsics and intersect it with the dome in the proxy 3D coordinate frame. The dome radius is set to enclose the reliable proxy geometry, with a fixed upper bound to avoid placing the texture excessively far from the camera. Each intersection point is converted to spherical coordinates and used to sample a 2D texture defined on the dome. The final dense conditioning image I_{cond} is obtained by compositing the source-derived warp I_{warp} with the rendered dome image I_{dome}:

$$I_{\text{cond}} = M \odot I_{\text{warp}} + (1 - M) \odot I_{\text{dome}}, \tag{3}$$

where $M \in \{0, 1\}^{H \times W}$ is the forward-warp validity mask; $M = 1$ selects source-derived warped pixels, while $M = 0$ selects the rendered dome.

We divide the 2D texture into a 16×16 grid and place a depth-3 recursive Sierpinski motif in each cell, alternating the triangle orientation across rows and columns. Rendered from the target camera pose, this spherical texture moves coherently with the camera, replacing the commonly used black background with a motion-dependent structured cue wherever the source-derived warp is invalid. The resulting conditioning video preserves source-derived content where available while supplying a geometrically consistent motion signal in newly revealed regions. Fig. 7 and Fig. 8 in appendix show examples of the resulting textured dome, and further implementation details are provided in Sec. B.1.

4.2 NegRoPE for Position-Disentangled Source Injection

The camera-controlling videos are spatially aligned with the generated output and can therefore be treated as aligned conditional signals. In contrast, the source video, which provides visual characteristics of the scene, is generally captured from a different viewpoint and lacks spatial correspondence with the output video, thus requiring a fundamentally different injection strategy. Rather than introducing a dedicated reference-attention module or relying on per-video adaptation, we use an architecture-preserving token-wise injection mechanism inspired by Flux Kontext [11]. We convert the source and target components into separate token sequences and concatenate them into a shared transformer sequence.

Let s_t denote the noised source video latent and s its clean counterpart. Similarly, let z_t denote the noised target video latent, and let c denote the clean camera-controlling video latent, which is spatially aligned with the target trajectory. We construct the source and target token sequences, x_t^{src} and x_t^{tar}, respectively, by patchifying the channel-wise concatenation of the corresponding components:

$$x_t^{\text{src}} = \text{Patchify}([s_t, s]_{\text{channel}}), \tag{4}$$

$$x_t^{\text{tar}} = \text{Patchify}([z_t, c]_{\text{channel}}). \tag{5}$$

We then concatenate them along the token dimension as $x_t = [x_t^{\text{src}}, x_t^{\text{tar}}]_{\text{token}}$. This allows them to jointly participate in self-attention and propagate reference information without requiring spatial alignment or fixed frame counts. This strategy works directly with the base model without any architectural modification.

However, if we assign positional embedding indices to the tokens in x_t^{src} and x_t^{tar} in the same way, performance drops significantly. This is because tokens with the same index are more likely to yield high dot product values, causing the model to incorrectly associate source and target tokens based on positional similarity rather than semantic content. To distinguish token types, we employ a simple yet effective trick: target tokens use positive spatial indices $n > 0$, while source tokens use negative spatial indices $-n$, with temporal ordering preserved within each sequence. A key property of RoPE makes this particularly elegant: the rotary embedding of a negative index $-n$ is equal to the complex conjugate of the rotary embedding of index n. Specifically, for queries and keys:

$$\bar{q}_{-n,c} = q_{-n,c} \exp(-i\omega_c n) = q_{-n,c} \exp(i\omega_c n)^*, \tag{6}$$

$$\bar{k}_{-n,c} = k_{-n,c} \exp(-i\omega_c n) = k_{-n,c} \exp(i\omega_c n)^*. \tag{7}$$

This encourages attention scores between target and reference tokens to incorporate relative distance while maintaining distinct positional signatures, reducing spurious positional correlations.

5 Experiments

5.1 Experimental Setup

Implementation details. We adopt Wan2.1 Fun-Control 14B [23] as our base model, using the 1.3B version for ablation study. Originally conditioned on text, first-frame images, and depth/human pose controls, the models are fine-tuned via LoRA (rank 64, lr 5×10^{-5}, 100 epochs) to adapt to our new control signals. To improve robustness, we randomly zero-out the first-frame signal with 10% probability. The models were trained and evaluated using NVIDIA L40S GPUs.

We train primarily on the MultiCamVideo [2] dataset, a synthetic collection rendered in Unreal Engine [5] where each scene features a 3D character captured by ten synchronized cameras. For each training sample, we randomly select two views: one acts as the source to construct the camera-controlling video, and the other serves as the ground-truth target. We generate two types of control signals: (1) sparse 3D point tracks via SpaTrackerV2 [26] and (2) dense point clouds derived from DepthAnything v3 [12]. To mitigate synthetic bias and preserve realistic appearance, we augment the training with real-world static scenes from RealEstate10K [32]. Control videos for this subset are derived from a single video using a first-frame point cloud projected onto subsequent frames. Crucially, this dataset is used without a reference source video, which enables the model to function effectively in scenarios where no reference is available. All inputs are resized and center-cropped to 512×320 pixels with 49 frames.

Evaluation protocol. We construct an evaluation set using 90 videos from the DAVIS dataset [19]. Following ReCamMaster [2], we use its 10 standard target camera trajectories and further add 4 more challenging trajectories, yielding 1,260 video-trajectory test cases in total. The complete set of target trajectories is visualized in the Sec. C.2. We evaluate each method with metrics for visual quality, geometric consistency, source preservation, and camera controllability. Specifically, we use VBench [8] for visual quality, Dyn-MEt3R [16] for geometric consistency of generated retakes, and per-frame MEt3R [1] for consistency with the input video. For camera controllability, we report Rotation Error (RotErr), Translation Error (TransErr), and Absolute Trajectory Error (ATE).

5.2 Comparison with State-of-the-Art Methods

We compare our method with recent camera-controlled video-to-video approaches: the implicit methods ReCamMaster [2] and ReDirector [17], and the explicit method TrajectoryCrafter [29].

Qualitative results. A known limitation of implicit methods is that they often conflate object and camera motion, causing dynamic objects to appear static or anchored during retaking. As shown across all examples in Fig. 4, ReCamMaster and ReDirector tend to keep objects visible even when they should leave the camera frustum; for example, the subject moves forward in the source video but remains nearly stationary in their retaken results. Although TrajectoryCrafter alleviates this issue with explicit guidance, it becomes unreliable when the guidance becomes sparse; for example, it hallucinates new objects after the original subject leaves the view or fails to sufficiently realize the specified camera trajectory. In contrast, SierpinskiCam preserves a clear separation between camera motion and object dynamics and produces context-consistent videos even under sparse geometric guidance.

Figure 4: **Qualitative comparison on the DAVIS dataset.** For each scene, top row shows the source video and target trajectory, where the camera moves from blue → red. Note how prior methods incorrectly put main characters (human/swan) at the center of the frames or create spurious artifacts, while SierpinskiCam accurately follows the user-defined camera paths. Left: the person should go down since the camera is fixed. Right: the swan should disappear from wide camera motion.

Table 1: **Quantitative comparison on the DAVIS dataset.** SierpinskiCam achieves the best or second-best performance on most metrics, with particularly large gains in geometric consistency and camera accuracy. **Bold**: best; underline: second best.

Method	Visual Quality↑					Geometric Consistency		Camera Accuracy		
	Subject Consistency	Background Consistency	Aesthetic Quality	Imaging Quality	Temporal Flickering	Dyn-MEt3R↑	MEt3R↓	RotErr↓	TransErr↓	ATE↓
ReCamMaster [2]	0.8983	**0.9170**	0.4875	0.4849	0.9636	0.7003	0.4562	0.2106	**1.2312**	0.7221
TrajectoryCrafter [29]	0.8625	0.9069	0.4883	0.4883	0.9439	0.7020	0.4421	0.2514	1.3091	0.7798
Redirector [17]	**0.9033**	0.9115	0.4943	**0.5186**	0.9614	0.7459	**0.3937**	0.2262	1.2537	0.7087
Ours (SierpinskiCam)	0.8986	0.9157	**0.4980**	0.5061	**0.9699**	**0.7477**	0.4097	**0.2058**	1.3087	**0.6921**

Quantitative results & User study. As reported in Tab. 1, SierpinskiCam performs favorably across the evaluated metrics, with the clearest improvements in camera accuracy and geometric consistency. The camera-accuracy gains suggest that the structured Sierpinski cues help the model follow the intended camera motion more faithfully. The user study further supports these findings from a perceptual perspective. As shown in Tab. 2-(c), our method achieves the highest mean score on all criteria, consistently outperforming other baselines. This study was conducted with 41 participants across 10 diverse DAVIS videos with varying camera motions, where participants rated each method on overall preference, camera motion accuracy, and source consistency using a 1–5 Likert scale.

5.3 Texture Design for Trackable Dome Conditioning

The dome texture is intended to provide auxiliary trackable structure for the target camera transformation, complementing real image cues that are preserved wherever forward warping yields valid RGB evidence. We therefore analyze which texture design best supports reliable feature tracking for dome conditioning. In particular, we compare structured patterns that provide different types of local visual cues: a Sierpinski triangle pattern with dense multi-scale edges and corners, a circle-fractal pattern with recursive smooth curves, a checkerboard pattern, a triangle grid, and a uniform textureless control. To evaluate this design choice, we render each candidate texture directly on the virtual dome under 10 camera trajectories from ReCamMaster, then extract SIFT features [14], match descriptors with Lowe's ratio test, and count geometrically consistent inliers using RANSAC [6]. Fig. 5 shows each candidate dome texture and summarizes its feature trackability.

This comparison isolates the structural cues that make a dome texture trackable. Although regular structured patterns such as the checkerboard and triangle grid produce detectable features, they yield fewer matches than the multi-scale patterns, indicating importance of multi-scale structure. Among the multi-scale variants, Sierpinski produces more verified inliers than the circle-fractal baseline,

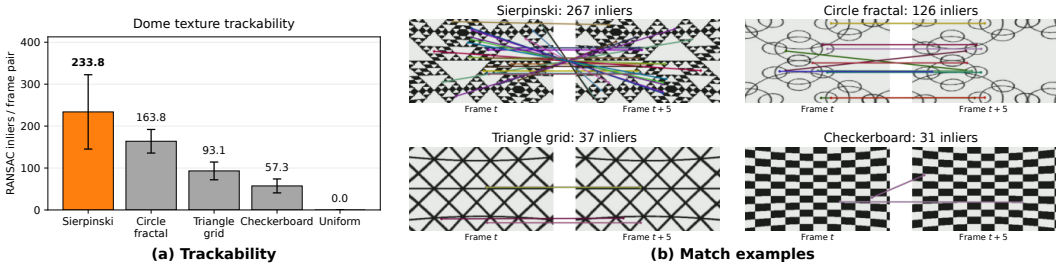

Figure 5: **(a)** Pattern-only trackability over 10 camera trajectories. Ours (Sierpinski) yields the most RANSAC-verified SIFT inliers per frame pair. **(b)** Representative frame-t to frame-$t{+}5$ matches; for clarity, only the top 10% inliers by Lowe ratio are drawn, while titles report total inliers.

Table 2: **Quantitative analysis of design choices and perceptual quality. (a)** Camera conditioning and texture design ablations show that combining texture and 3D information improves camera controllability, with the Sierpinski texture providing the strongest texture-based control. **(b)** Source-video injection ablations show that negative RoPE indexing best preserves generation quality by avoiding positional collisions between source and target tokens. **(c)** User study on DAVIS shows that participants prefer SierpinskiCam in overall quality, camera motion accuracy, and source stability.

(a) Conditioning and texture design

Study	Variant	Hard Set		Moderate Set	
		RotErr↓	TransErr↓	RotErr↓	TransErr↓
Conditioning	Plücker	0.024	0.042	0.021	0.036
	Texture Only	0.029	0.055	0.024	0.049
	3D Only	0.023	0.035	0.019	0.028
	Tex. + 3D	**0.022**	**0.034**	**0.018**	**0.027**
Texture	Checkerboard	0.025	0.042	0.021	0.041
	Sierpinski	**0.022**	**0.034**	**0.018**	**0.027**

(b) Source-video injection

Method	CLIP↑	DINO↑	FID↓
(a) Frame concat	0.816	0.612	164.87
(b) No RoPE mod.	0.842	0.651	162.83
(c) Offset RoPE	0.827	0.659	164.16
(d) Negative RoPE	**0.858**	**0.692**	**161.20**

(c) User Study

Method	User Study		
	Overall↑	Motion↑	Stab.↑
ReCamMaster [2]	2.70	3.17	2.72
TrajectoryCrafter [29]	2.56	3.12	2.44
ReDirector [17]	2.80	3.22	2.88
Ours (SierpinskiCam)	**3.23**	**3.33**	**3.32**

suggesting that high-contrast triangular edges and corners provide stronger local features than smooth circular structures. Based on this evidence, we adopt the Sierpinski triangle as the default dome texture. This provides dense, geometry-verifiable cue in dome-conditioned regions, giving the model additional trackable structure beyond the warped RGB signal.

5.4 Ablation Study

Camera conditioning signal. We first compare our proposed hybrid camera conditioning signal with the standard Plücker ray representation. To strictly evaluate camera controllability, we use a dedicated setting trained on 1,254 static RealEstate10K [32] sequences and evaluated on 150 static scenes. This static-scene protocol removes the confounding effects of object motion, allowing us to directly assess how faithfully each conditioning signal controls the generated camera trajectory. Test scenes are stratified into hard (top 30% motion intensity) and moderate sets, with metrics computed using camera poses estimated via VGGT [24]. Tab. 2-(a) reports the results on this split. The full signal, which combines texture and 3D information, achieves the strongest performance across all metrics and difficulty levels, clearly outperforming Plücker rays. While texture-only and 3D-only variants already improve over the baseline, combining the two provides further consistent gains, demonstrating the complementarity between dense texture flow and explicit geometric anchors.

Next, we examine the texture component in isolation to validate the design factors studied in Sec. 5.3. Using the same controlled RealEstate10K protocol, we compare our Sierpinski texture with a checkerboard baseline, which can be viewed as an ablated texture that lacks the key multi-scale, corner-rich structure of our design. Tab. 2-(a) summarizes the results under this texture-only setting. In line with the earlier analysis, the Sierpinski texture consistently reduces both rotational and translational errors compared to the checkerboard.

Table 3: **Further analysis on MultiCamVideo and generalization of Sierpinski dome to another method. (a)** On MultiCamVideo, our method achieves the best performance across reconstruction, perceptual, semantic, and distribution-level metrics. **(b)** On DAVIS, applying our method to ReAngle-A-Video improves camera-pose metrics over the original approach. **Bold**: best.

(a) MultiCamVideo comparison

Method	PSNR↑	LPIPS↓	CLIP↑	DINO↑	FID↓
ReCamMaster [2]	17.6743	0.4106	0.9247	0.8727	33.2497
ReDirector [17]	17.9585	0.3807	0.9282	0.8743	33.1327
Ours (SierpinskiCam)	**19.6368**	**0.2923**	**0.9378**	**0.8796**	**32.8053**

(b) Camera accuracy in the ReAngle-A-Video setting

Method	RotErr↓	TransErr↓	ATE↓
ReAngle-A-Video	**0.1591**	1.3968	2.1409
w/ Sierpinski	0.1613	**1.3087**	**1.6015**

Source video injection strategies. Finally, we evaluate source injection strategies on the held-out test split of the MultiCamDataset [2], where ground-truth target views are available, using the Wan 1.3B model. We compare frame concatenation (baseline) against three token-concatenation schemes: (a) unmodified RoPE indices (identical source/target positions), (b) spatially offset RoPE indices (+4096 pixel displacement), and (c) our proposed NegRoPE indices. As shown in Tab. 2-(b), the NegRoPE strategy consistently outperforms all baselines. Unlike offsets, negative indexing guarantees zero positional collision between source and target tokens. This performance advantage holds across feature-based metrics (CLIP [20], DINO [15]) as well as pixel-level metrics (PSNR, LPIPS) with first-frame conditioning.

5.5 Further Analysis

MultiCamVideo Evaluation. To complement potentially ambiguous generative video metrics, we additionally evaluate on a held-out set of 100 MultiCamVideo scenes [2], which provide ground-truth videos under known camera trajectories. [3] This controlled setting allows us to directly measure how closely each method matches the target-view video. As shown in Tab. 3-(a), our method achieves overall favorable results against the MultiCamVideo-adapted baselines across these metrics.

Extension to an explicit camera-control method. To examine whether our structured camera cue can extend beyond our own pipeline, we instantiate it in an existing explicit method. Specifically, we use ReAngle-A-Video [9], which provides training code. We follow the original codebase and modify only the construction of the conditioning video used for optimization. We evaluate this extension on 7 DAVIS scenes with 5 target camera trajectories. The results in Tab. 3-(b) suggest that the structured cue can improve trajectory adherence, particularly in translation and ATE. This experiment shows the plug-in applicability of the proposed structured camera cue to an explicit camera-control pipeline beyond our own implementation.

6 Conclusion and Limitations

In this work, we presented SierpinskiCam, a dynamic video retaking framework that combines reconstructed 4D geometry with Sierpinski-patterned dome conditioning. The Sierpinski dome provides trackable multi-scale structure that complements warped RGB evidence and helps guide the target camera motion. Our analyses validate this design and confirm its benefit for video model conditioning. We further introduce NegRoPE to disentangle source and target positions, allowing unaligned source videos to guide generation without spurious positional matching. Results show that SierpinskiCam maintains strong camera control, geometric consistency, and visual fidelity across diverse scenarios, including large camera deviations where little original scene evidence remains available. As an inherent consequence of relying on video generation and 4D estimation models, SierpinskiCam also inherits their limitations. For example, complex motion in original video or unreliable 4D reconstruction can lead to degraded results. Future advances in both model families may help mitigate these limitations.

[3] Our method uses MultiCamVideo for training, but the evaluated 100 scenes are held out. In contrast, these scenes may overlap with the training data of ReCamMaster and ReDirector. We exclude TrajectoryCrafter, which is not trained on MultiCamVideo.

References

[1] Mohammad Asim, Christopher Wewer, Thomas Wimmer, Bernt Schiele, and Jan Eric Lenssen. Met3r: Measuring multi-view consistency in generated images. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6034–6044, 2025.

[2] Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, et al. Recammaster: Camera-controlled generative rendering from a single video. *arXiv preprint arXiv:2503.11647*, 2025.

[3] Wei Cao, Hao Zhang, Fengrui Tian, Yulun Wu, Yingying Li, Shenlong Wang, Ning Yu, and Yaoyao Liu. Freeorbit4d: Training-free arbitrary camera redirection for monocular videos via geometry-complete 4d reconstruction. *arXiv preprint arXiv:2601.18993*, 2026.

[4] Kaihua Chen, Tarasha Khurana, and Deva Ramanan. Reconstruct, inpaint, finetune: Dynamic novel-view synthesis from monocular videos. *arXiv preprint arXiv:2507.12646*, 2025.

[5] Epic Games. Unreal engine. URL `https://www.unrealengine.com`.

[6] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. doi: 10.1145/358669.358692.

[7] Google DeepMind. Veo 3. `https://deepmind.google/models/veo/`, 2025. Model card, accessed 25/Jul/2025.

[8] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024.

[9] Hyeonho Jeong, Suhyeon Lee, and Jong Chul Ye. Reangle-a-video: 4d video generation as video-to-video translation. *arXiv preprint arXiv:2503.09151*, 2025.

[10] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[11] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

[12] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[13] Kuan Heng Lin, Zhizheng Liu, Pablo Salamanca, Yash Kant, Ryan Burgert, Yuancheng Xu, Koichi Namekata, Yiwei Zhao, Bolei Zhou, Micah Goldblum, et al. Vista4d: Video reshooting with 4d point clouds. *arXiv preprint arXiv:2604.21915*, 2026.

[14] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94.

[15] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[16] Byeongjun Park, Hyojun Go, Hyelin Nam, Byung-Hoon Kim, Hyungjin Chung, and Changick Kim. Steerx: Creating any camera-free 3d and 4d scenes with geometric steering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27326–27337, 2025.

[17] Byeongjun Park, Byung-Hoon Kim, Hyungjin Chung, and Jong Chul Ye. Redirector: Creating any-length video retakes with rotary camera encoding. *arXiv preprint arXiv:2511.19827*, 2025.

[18] Jangho Park, Taesung Kwon, and Jong Chul Ye. Zero4d: Training-free 4d video generation from single video using off-the-shelf video diffusion. *arXiv preprint arXiv:2503.22622*, 2025.

[19] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

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

[21] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[22] 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 *European Conference on Computer Vision*, pages 313–331. Springer, 2024.

[23] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[24] 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.

[25] Kevin Wooley and Ronald Mallet. Scale independent tracking pattern. U.S. Patent US9672417B2, June 2017. URL `https://patents.google.com/patent/US9672417B2/en`. Assigned to Lucasfilm Entertainment Co. Ltd.

[26] Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Yuri Makarov, Bingyi Kang, Xing Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. Spatialtrackerv2: 3d point tracking made easy. *arXiv preprint arXiv:2507.12462*, 2025.

[27] Zeqi Xiao, Wenqi Ouyang, Yifan Zhou, Shuai Yang, Lei Yang, Jianlou Si, and Xingang Pan. Trajectory attention for fine-grained video motion control. *arXiv preprint arXiv:2411.19324*, 2024.

[28] 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. *arXiv preprint arXiv:2408.06072*, 2024.

[29] Mark YU, Wenbo Hu, Jinbo Xing, and Ying Shan. Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. *arXiv preprint arXiv:2503.05638*, 2025.

[30] David Junhao Zhang, Roni Paiss, Shiran Zada, Nikhil Karnad, David E Jacobs, Yael Pritch, Inbar Mosseri, Mike Zheng Shou, Neal Wadhwa, and Nataniel Ruiz. Recapture: Generative video camera controls for user-provided videos using masked video fine-tuning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2050–2062, 2025.

[31] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

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

A Motivation

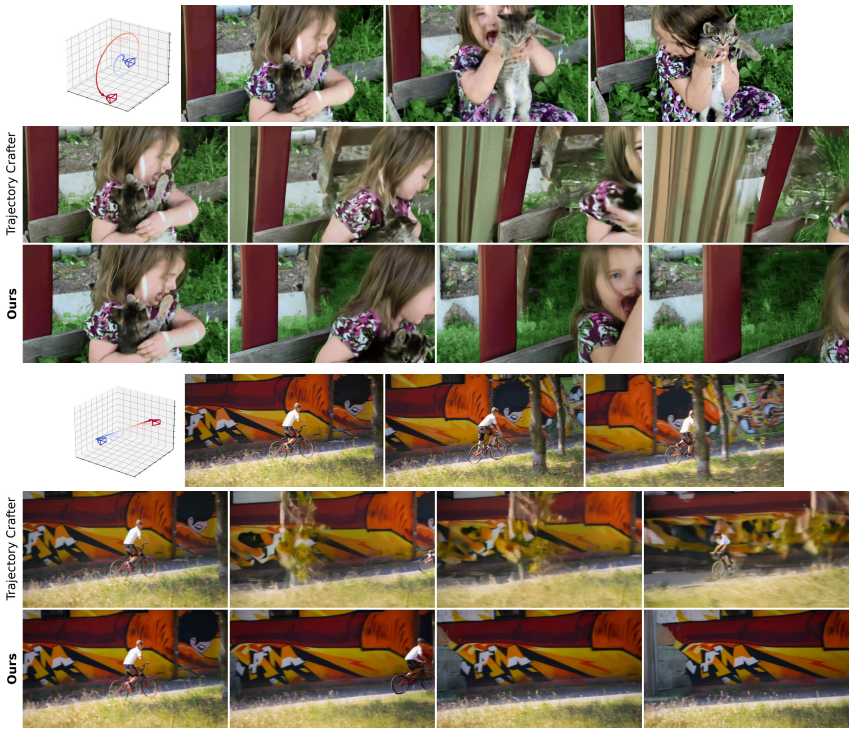

Figure 6: **Failure cases of TrajectoryCrafter.** When the target camera moves beyond the original scene coverage, it may hallucinate content or fail to follow the camera pose, especially as the main object becomes small or leaves the target-view frustum.

We observed that TrajectoryCrafter [29] often fails when the target camera trajectory extends far beyond the original scene coverage. As shown in Fig. 6, when the camera moves away from the main object or the object leaves the target-view frustum, the model hallucinates content or fails to follow the given camera pose. We attribute this partly to the common practice of rendering the proxy over a black background: when the warped RGB signal covers only a small part of the target view, the remaining regions provide little cue about depth or camera motion. We therefore add a Sierpinski-textured dome to the proxy rendering, providing trackable multi-scale structure in empty regions and making the target camera motion more explicit.

B Implementation Details

B.1 Textured Dome Generation

We provide additional details on the construction of the textured dome used for the dense conditioning signal. The dome is implemented as a sphere in the proxy 3D coordinate frame. For each target camera, we cast one ray per output pixel using the target intrinsics and intersect the ray with the sphere. The intersection point is converted to spherical coordinates, and its latitude and longitude are used to sample a 2D texture image. In our implementation, the sphere radius is set as

$$R = \min(R_{\max}, d_{\max}), \tag{8}$$

where $d_{\max}$ is the maximum distance of confident points in the estimated proxy geometry and $R_{\max} = 30$. This clipping prevents the dome from being placed excessively far from the camera while still enclosing the reliable proxy geometry. In fixed-radius analyses, we directly set $R = 30$.

The final dense conditioning frame is obtained by compositing the forward-warped source proxy with the rendered dome:

$$I_{\text{cond}} = M \odot I_{\text{warp}} + (1 - M) \odot I_{\text{dome}}, \tag{9}$$

where I_{warp} is the source-derived target-view proxy, I_{dome} is the rendered textured dome, and $M \in \{0,1\}^{H\times W}$ is the forward-warp validity mask. Pixels with $M = 1$ use the warped source content, while pixels with $M = 0$ are filled by the dome.

The default dome texture is a tiled Sierpinski-triangle pattern. We generate a 2048×2048 RGB texture and divide it into a 16×16 grid. Each grid cell contains a recursive Sierpinski triangle with recursion depth 3. The triangle orientation alternates across rows and columns, and additional half-cell-shifted triangles are inserted according to the row/column parity to avoid large texture regions with a single dominant orientation. This produces a repeated multi-scale pattern of corners and edges across the entire sphere.

Concretely, let s denote the grid-cell size. For each cell, we recursively subdivide a triangle of size s into three child triangles of size $s/2$ until depth 3 is reached, at which point the leaf triangles are rasterized into the texture. The orientation of the root triangle is alternated by row index, and half-cell-shifted triangles with the opposite orientation are added on alternating columns. This staggered tiling reduces large empty bands and keeps local high-contrast structures visible under different target camera views. Fig. 7 shows the Sierpinski triangle pattern used as our geometry proxy alongside conditioning video examples across far-field and near-field scenes. Triangle size varies with the scene depth (smaller for far-field and larger for near-field), providing the model with spatial depth cues.

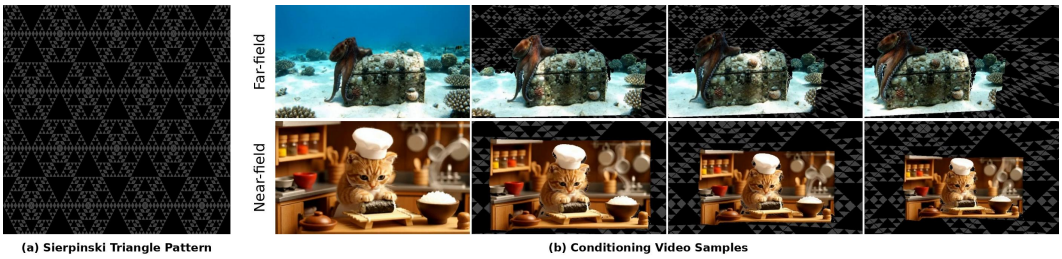

Figure 7: **Geometry proxy design and conditioning video examples. (a)** The Sierpinski triangle pattern used as a geometry proxy for camera motion conditioning. Its self-similar structure provides rich spatial cues without scene-specific texture, enabling generalizable control. **(b)** Example conditioning videos under two representative depth regimes: far-field (distant background) and near-field (close background), illustrating the range of depth variation handled by our framework.

C Experimental Details

C.1 Caption Preparation

The videos used in our experiments are automatically captioned using CogFlorence[4], a fine-tuned variant of Microsoft's Florence-2 model, and the resulting captions are used for downstream evaluation.

C.2 Additional Evaluation Trajectories

To evaluate robustness under more challenging camera motions, we augment the 10 standard ReCamMaster trajectories with 4 additional target trajectories. These trajectories are designed to stress large viewpoint changes and out-of-frame extrapolation, including large panning, zoom-out, and whirl motions. Figure 8 visualizes the added trajectories used in our DAVIS evaluation.

C.3 Camera Pose Evaluation

We evaluate camera paths by first estimating camera trajectories from video sequences and then comparing them against ground-truth camera paths. For static scenes, we estimate camera poses using VGGT [24], which provides stable and accurate camera motion under static-scene assumptions. For dynamic scenes, we adopt DepthAnythingV3 (DA3) [12] to infer depth and recover camera motion.

[4] `https://huggingface.co/thwri/CogFlorence-2.2-Large`

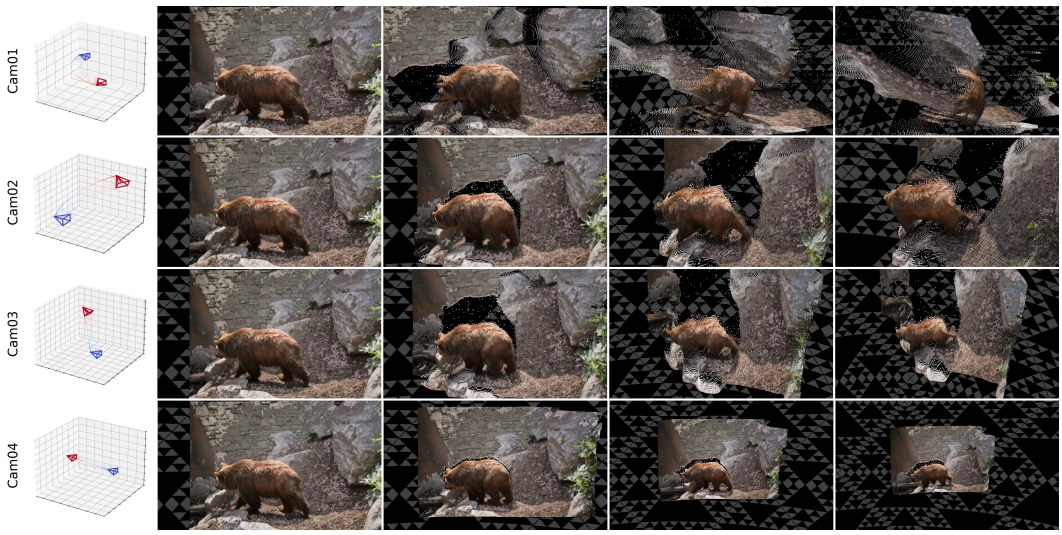

Figure 8: **Additional camera trajectories used for evaluation.** Each row shows the geometry proxy frames used as video conditioning input for four additional camera trajectories. The sampled frames illustrate the spatial extent and viewpoint variation induced by each trajectory.

Although DA3 represents one of the latest advances in dynamic camera motion estimation, we observe that its estimated camera paths still contain significant inaccuracies, even when applied to static scenes. To mitigate this issue, we concatenate five independently generated videos from the same scene into a single longer video sequence and jointly estimate the camera trajectory. Empirically, this strategy substantially improves camera path estimation accuracy.

C.4 User Study

For user study, participants were recruited through Prolific[5] and were paid approximately $4 each. They were shown the source video, an explanation of the target camera trajectory, and four retaking results. The four candidate videos were randomly ordered and labeled as A, B, C, and D for each question. Participants were then asked to rate each candidate independently using the following instructions.

C.5 Camera Motion Metric for Test Set Stratification

In Sec. 5.4, we define a Motion Intensity Score (MIS) to quantify the difficulty of camera motion in each test sequence. Since conditional view synthesis becomes more challenging as the camera moves farther from the reference pose, MIS summarizes this difficulty by aggregating translational and rotational motion along the camera trajectory.

$$MIS = \sum_{i=1}^{n-1} \sqrt{\|\mathbf{t}_{i+1} - \mathbf{t}_i\|_2^2 + \lambda |\Delta\theta_i|^2}, \tag{10}$$

where $\mathbf{t}_i$ denotes the camera translation at frame i, $\Delta\theta_i$ is the geodesic distance between consecutive camera rotations, and λ is a weighting factor balancing translational and rotational motion. We set $\lambda = 2.0$. By accumulating frame-to-frame motion, MIS reflects both the magnitude and complexity of camera trajectories. We rank test sequences by MIS and use this ranking to stratify the test set into different difficulty levels.

C.6 MulticamVideo Evaluation Details.

All camera-controlling videos in the MultiCamVideo evaluation follow the training setup: 49 frames at 832×480 resolution. TrajectoryCrafter uses point-cloud controls without textured backgrounds,

[5]https://www.prolific.com/

User study instructions:
Please rate each result from 1 to 5 (1 = Very poor, 5 = Excellent) based on three criteria:

1. Overall preference
Please rate each result based on your overall preference, considering visual quality, realism, temporal coherence, and similarity to the source video.

2. Camera motion accuracy
Please rate each result based on how well its camera motion follows the target trajectory described in the question.

3. Stability & source consistency
Please rate each result based on temporal stability and consistency with the source video. A higher score means less flickering, fewer unexpected changes in the subject/background, and better preservation of the source identity and geometry.

Table 4: Instructions used for the user study.

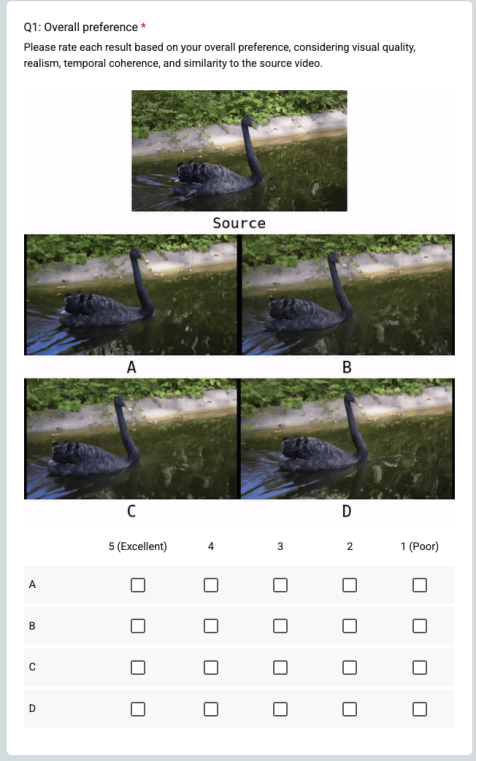

Figure 9: Screenshot of the user study interface.

while ReCamMaster and ReDirector use ground-truth camera poses. We compare the generated videos against the ground-truth videos using pixel-wise and feature-based metrics, and report FID to measure distribution-level similarity.

C.7 ReAngle-A-Video Extension Details.

We extend ReAngle-A-Video to the DAVIS 480p setting and evaluate whether the generated videos preserve the intended camera motion. For each DAVIS sequence, we use the first 49 frames and resize/crop them to the ReAngle training resolution of 720×480. We evaluate five camera trajectories: left, right, up, down, and zoom-out. For the vanilla baseline, we use the default ReAngle-A-Video conditioning pipeline. For our variant, we replace the black disocclusion fill with the proposed Sierpinski-pattern background while keeping the same target camera trajectory and Depth Anything 3 (DA3) -estimated scene geometry. Both methods use DA3 depth and intrinsics for constructing the conditioning videos. We use 150 optimization steps. To quantify camera-motion adherence, we estimate camera poses from the generated videos using DA3. We then compare the estimated trajectory against the target camera trajectory. Rotation error is reported in radians and is computed relative to the first frame to reduce sensitivity to fixed camera-coordinate convention differences.

D Additional Qualitative Results

We provide additional qualitative results and comparisons beyond the main evaluation dataset. These results further demonstrate that SierpinskiCam generalizes across different data sources and camera-motion scenarios. We also include supplementary videos to better visualize camera-control behavior and the temporal dynamics.

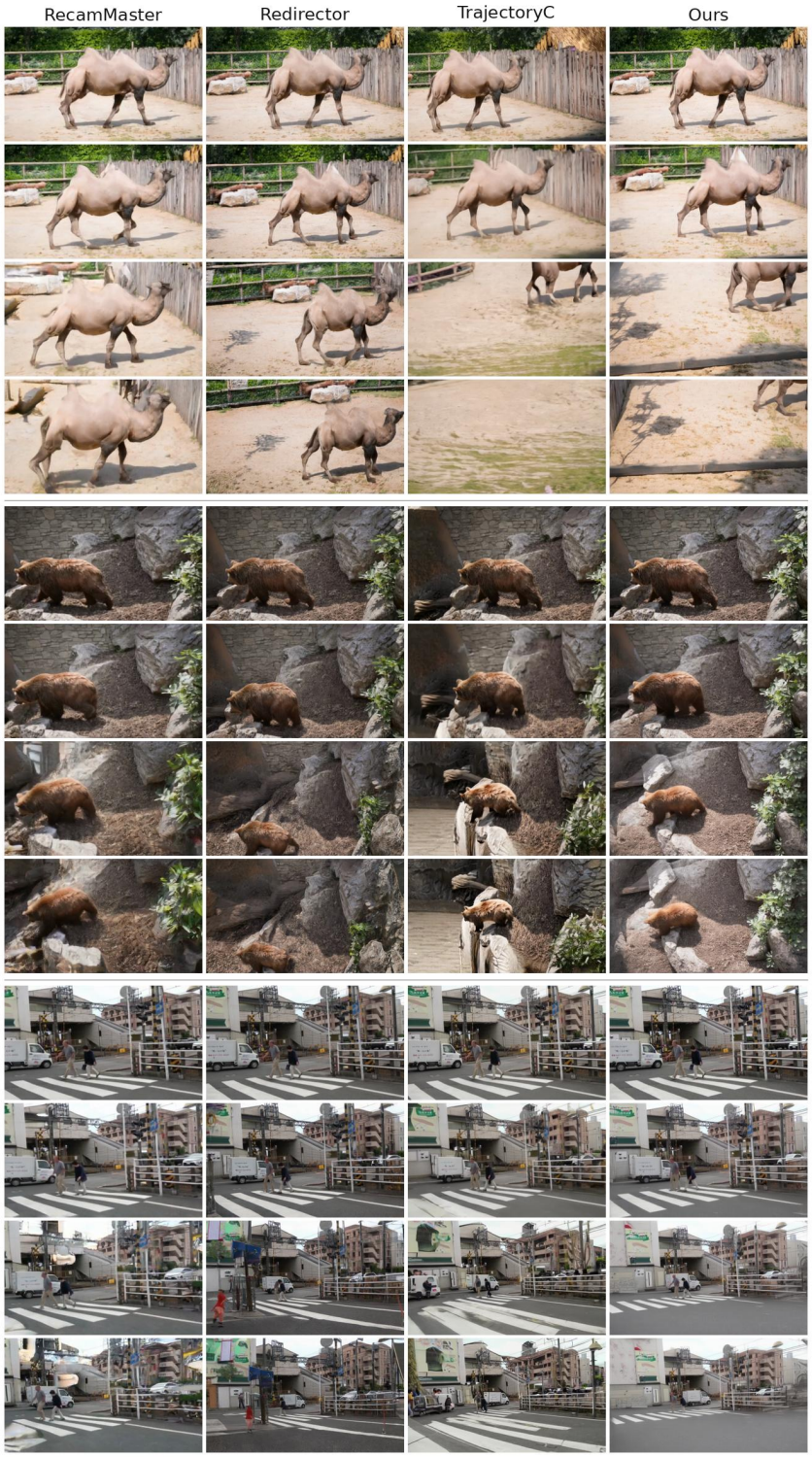

Figure 10: **Additional comparison on newly added challenging camera trajectories.**

Figure 11: **Additional qualitative comparison on the DAVIS dataset**.

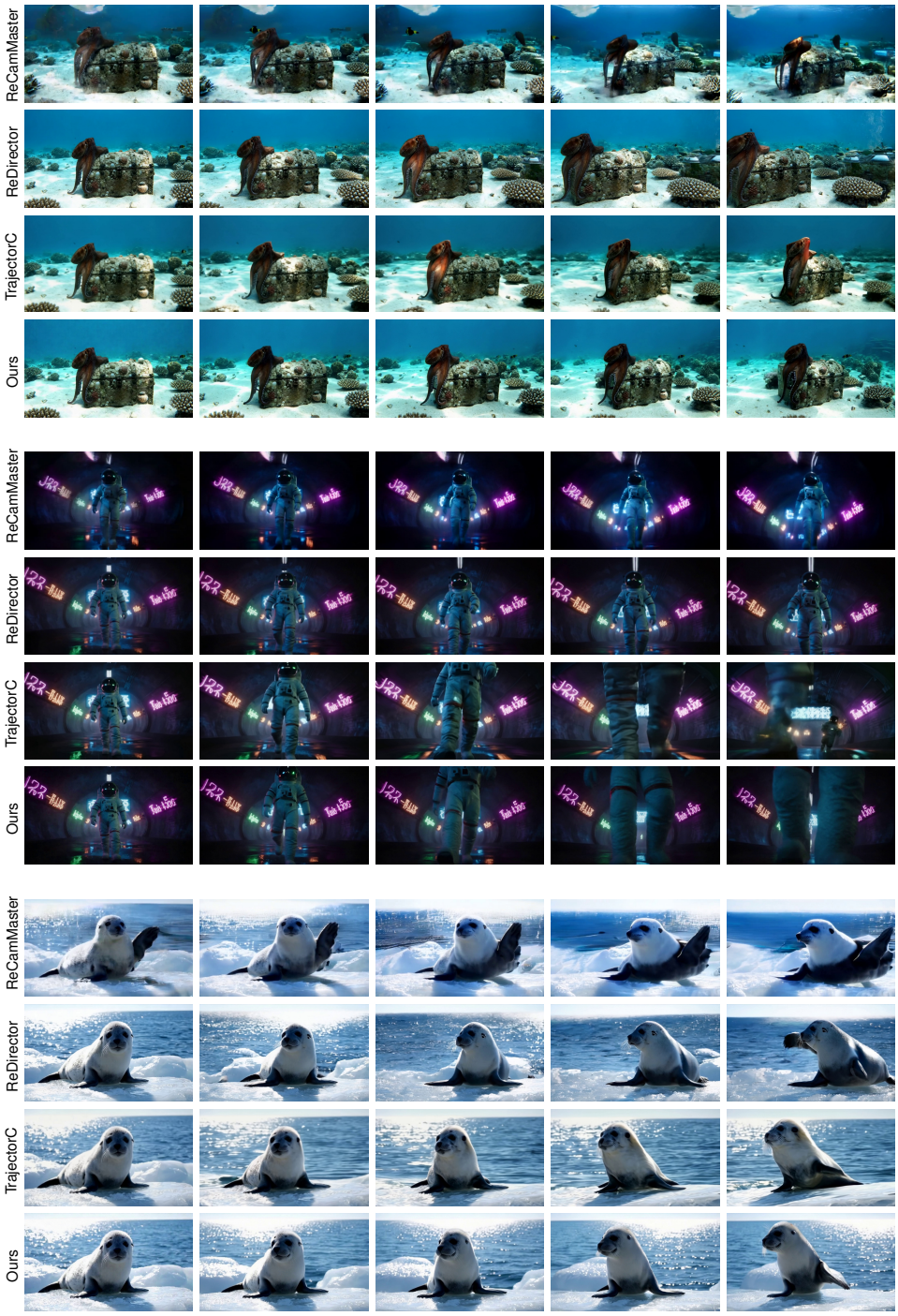

Figure 12: **Additional qualitative comparison on generated video by Veo**.

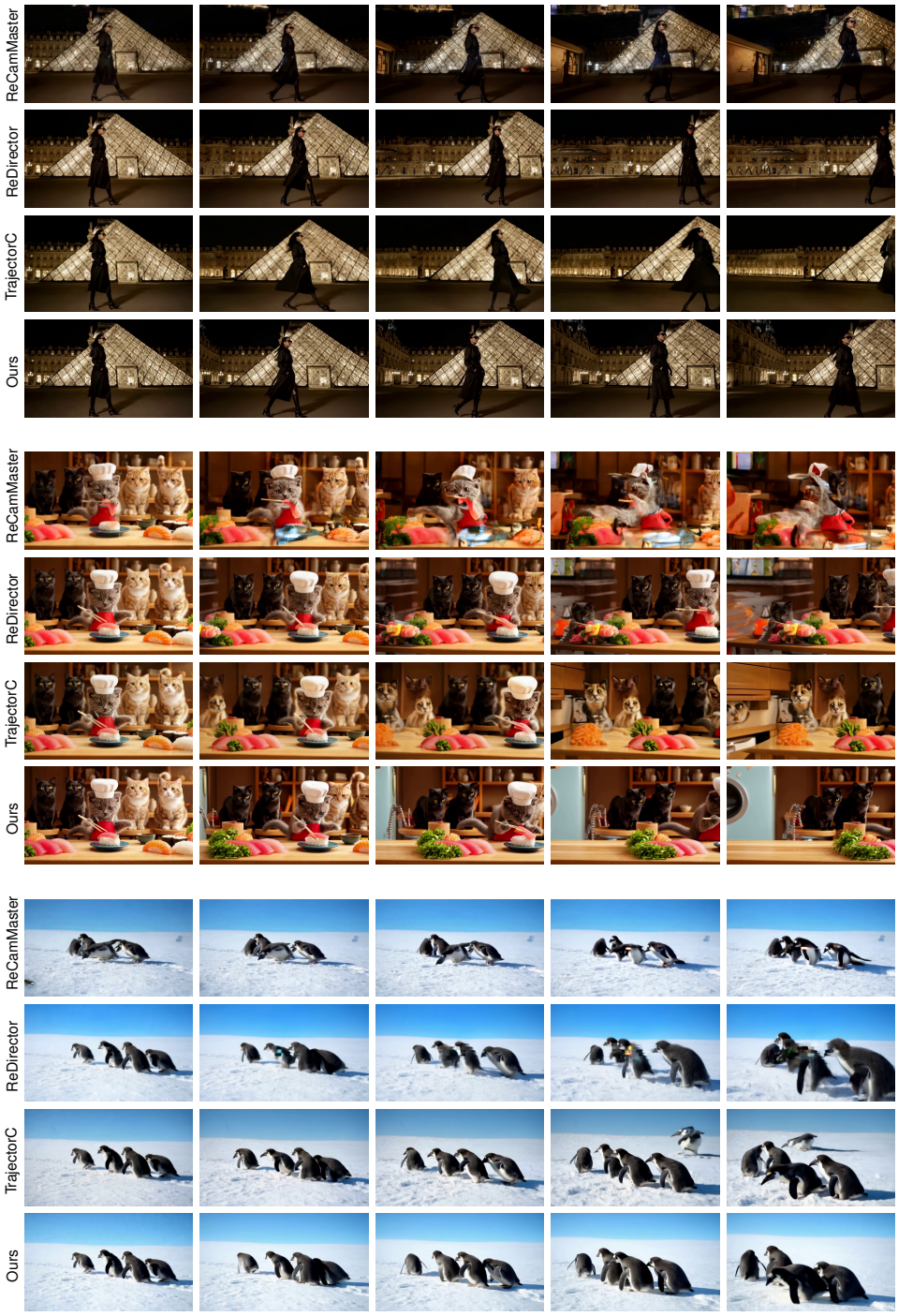

Figure 13: **Additional qualitative comparison on generated video by Veo**.

E Societal Impact

As this work involves generative models, it may have both positive and negative societal impacts. Potential positive impacts include supporting creative and scientific applications, while potential negative impacts include misuse for generating misleading or harmful content. We encourage responsible use and further assessment before deployment in sensitive domains.

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: The abstract and introduction summarize the proposed method, evaluation setting, and main empirical findings. The claims are limited to the contributions and experimental evidence presented in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We include the limitations in the conclusion part.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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 the necessary implementation and experimental details in the main text and Appendix to reproduce the main experimental results supporting our claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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: The code are not yet public, but will be released.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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) necessary to understand the results?

   Answer: [Yes]

   Justification: We do provide the necessary implementation and experimental details in the main text and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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: [Yes]

   Justification: The bar plots report the mean values together with their standard deviation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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 specify that the experiments were conducted using NVIDIA L40S GPUs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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 do not identify any aspects of this work that violate the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss potential positive and negative societal impacts, noting that generative models may be beneficial for creative and scientific applications but may also pose risks of misuse.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release data or models with a high risk for misuse that would require additional safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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: We use publicly available open-source datasets and do not release data or models with a high risk for misuse that would require additional safeguards.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not introduce or release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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: We provide the participant instructions, study interface screenshots, and compensation details for the user study in the Appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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: [N/A]

    Justification: The user study was limited to anonymous perceptual ratings of generated videos and did not collect personally identifiable information. We followed the applicable institutional guidance for such minimal-risk studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only as writing assistants for grammar and expression refinement, and did not affect the core methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.