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

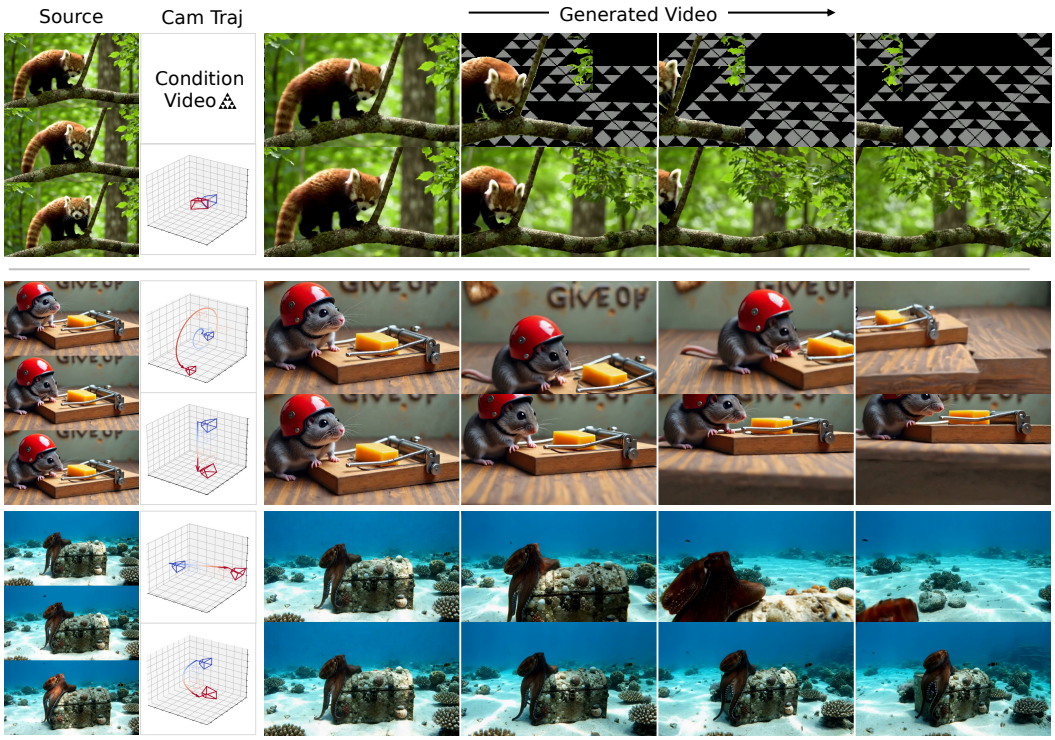

Figure 1: **SierpinskiCam for video retake generation.** Given a source video and a target camera trajectory (blue → red), SierpinskiCam retakes the video under user-defined camera motions. Even under large viewpoint changes with sparse source evidence, our Sierpinski textured dome (top *condition video*) 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

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

viewpoint changes. We further 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. Project page: https://hyelinnam.github.io/SierpinskiCam/.

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) [11, 24, 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, 19, 23] 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, 10, 20, 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 [23], 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 and Tab. 1), with 15% higher user preference scores (Tab. 2).

2 Related work

2.1 Camera Control for Video Retaking

Building on recent advances in video diffusion models (VDMs) [11, 24, 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, 19, 23] 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 [19] 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, 10, 20, 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, 15] 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 [23] 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, 10, 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 [22] 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

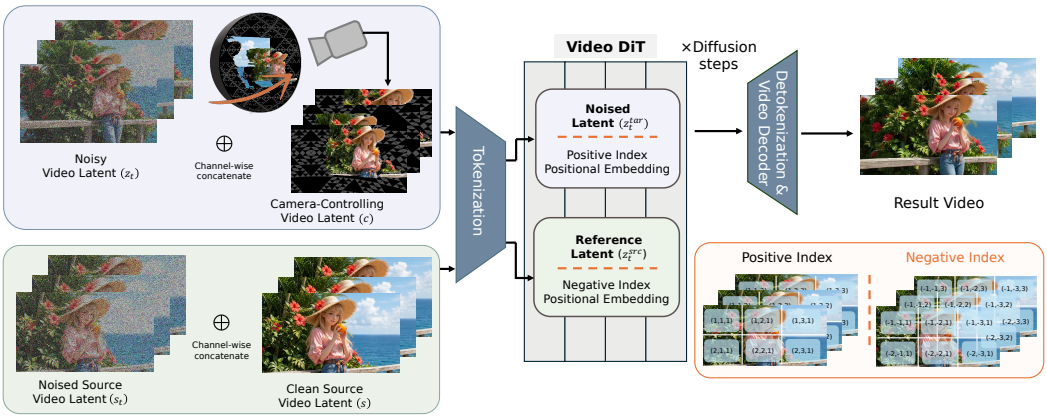

Figure 2: **Overview of SierpinskiCam**. Our model processes two parallel streams: (1) a target stream (top) where noisy target latents z_t are concatenated with the Sierpinski-dome camera-controlling video c, using positive RoPE indices for spatial alignment; and (2) a source stream (bottom) where noised 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 [24], 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

$$A'(n,m) = \text{Re}\left[\sum_{c=1}^{d_{\text{head}}/2} q_{n,c}\, k^*_{m,c}\, \exp\big(i\omega_c(n-m)\big)\right]. \tag{2}$$

RoPE makes the attention scores depend on token positions only through the relative offset $(n-m)$, 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 [14]

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.

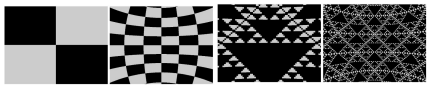

Figure 3: **Multi-scale robustness.** Unlike the Checkerboard pattern (a), the Sierpinski fractal pattern (b) provides structural details in both near and far views, ensuring camera pose control across scales.

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 [12]. 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 [24] 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 [14]. 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 evaluate our method on two complementary benchmarks. First, following prior video-retaking works [2], we evaluate on 90 videos from DAVIS [21], using the 10 standard target camera trajectories from ReCamMaster and 4 additional challenging trajectories, yielding 1,260 video–trajectory test cases in total. DAVIS provides diverse in-the-wild dynamic scenes and serves as our primary benchmark for evaluating retaking quality under realistic object and scene motion. We additionally evaluate on 56 clips from the Charge dataset [17], which provides synchronized target-view videos and ground-truth camera trajectories. This setting provides an independent benchmark for evaluating camera controllability and geometric consistency with known target cameras.

Across both benchmarks, we report VBench [9] metrics for visual quality, Dyn-MEt3R [18] for the geometric consistency of generated retakes, and per-frame MEt3R [1] for consistency with the source video. On Charge, where ground-truth target videos are available, we additionally report PSNR and LPIPS. We treat these metrics as complementary, since newly synthesized regions may plausibly differ from the ground truth. For camera controllability, we report Rotation Error (RotErr), Translation Error (TransErr), and Absolute Trajectory Error (ATE). Generated-video camera trajectories are estimated using ViPE [8], selected through validating three pose estimators on Charge (Appendix C.3).



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 leave the frame as the camera moves along the target trajectory.

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 [19], and the explicit method TrajectoryCrafter [29] and Vista4D [15].

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. Vista4D likewise employs explicit 4D point-cloud guidance, but still exhibits a subject-following bias in challenging cases. In the snowboard example, its synthesized camera partially follows the moving subject. This behavior is consistent with a conflict between its explicit point-cloud guidance and implicit source-video prior, which favors subject preservation over strict adherence to 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.

Quantitative results & User study. As reported in Tab. 1, SierpinskiCam performs favorably across both DAVIS and Charge, with particularly consistent performance in geometric consistency and camera controllability. The results on Charge, which provides ground-truth camera trajectories, further support the effectiveness of our structured camera cues for following the intended target motion under a controlled evaluation setting. The user study further supports these findings from a perceptual perspective. As shown in Tab. 2 our method achieves the highest mean score

Table 2: **User study on DAVIS.** Higher scores indicate stronger preference.

Method	User Study		
	Overall↑	Motion↑	Stab.↑
ReCamMaster [2]	2.70	3.17	2.72
TrajectoryCrafter [29]	2.56	3.12	2.44
ReDirector [19]	<u>2.80</u>	<u>3.22</u>	<u>2.88</u>
Ours (SierpinskiCam)	**3.23**	**3.33**	**3.32**

Table 1: **Quantitative comparison on DAVIS and Charge.** We evaluate recent camera-controlled video retaking methods across visual quality, geometric consistency, and camera accuracy. DAVIS provides an in-the-wild dynamic-video evaluation, while Charge provides synchronized target views and ground-truth camera trajectories. **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↓
DAVIS	ReCamMaster	0.8983	**0.9170**	0.4875	0.4849	0.9636	0.7003	0.4562	0.2106	**1.2312**	0.7221
	ReDirector	**0.9033**	0.9115	0.4943	0.5186	0.9614	0.7459	0.3937	0.2262	1.2537	0.7087
	TrajectoryCrafter	0.8625	0.9069	0.4883	0.4883	0.9439	0.7020	0.4421	0.2514	1.3091	0.7798
	Vista4D	0.8819	0.9153	0.4726	**0.5227**	0.9463	0.7323	**0.3113**	0.3249	1.3443	0.7253
	Ours	0.8986	0.9157	**0.4980**	0.5061	**0.9699**	**0.7477**	0.4097	**0.2058**	1.3087	**0.6921**
CHARGE	ReCamMaster	0.9134	0.9338	0.4950	0.4397	**0.9904**	0.7978	0.4380	11.3046	0.4391	0.3580
	ReDirector	0.9210	**0.9514**	0.4874	0.5358	0.9803	0.7996	**0.2665**	12.4306	**0.3144**	0.2991
	TrajectoryCrafter	0.9204	0.9512	0.5197	0.4413	0.9815	0.8291	0.3699	13.6855	0.3893	0.2985
	Vista4D	0.9168	0.9490	0.5243	0.5430	0.9767	0.8231	0.3056	13.8536	0.3873	0.2968
	Ours	**0.9226**	0.9462	**0.5466**	**0.5568**	0.9812	**0.8445**	0.2957	**11.2907**	0.3165	**0.2889**

across all criteria among the evaluated methods. The study compares SierpinskiCam with ReCamMaster, TrajectoryCrafter, and ReDirector on 10 diverse DAVIS videos with varying camera motions. A total of 41 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. We compare six patterns and a uniform baseline. The textured patterns are the proposed Sierpinski triangle, a recursive circle fractal, random dots, a triangle grid, Perlin noise, and a checkerboard. To isolate the effect of the pattern itself, we render each texture directly on the virtual dome without scene content or foreground compositing, using 10 canonical camera trajectories from ReCamMaster. We extract SIFT features [16], match descriptors using Lowe's ratio test, and perform RANSAC-based geometric verification [6]. For each trajectory, we average the number of verified inliers over adjacent-frame pairs and report the mean and standard deviation across trajectories.

Fig. 5 summarizes the resulting pattern trackability and shows representative matches under a larger temporal gap. The Sierpinski pattern achieves the highest mean number of verified correspondences, with 233.8 inliers per frame pair, followed by the circle-fractal pattern with 163.8. Random dots, the triangle grid, Perlin noise, and the checkerboard yield 115.9, 93.1, 60.2, and 57.3 inliers, respectively. Thus, the two recursive-multi-scale patterns provided more verified correspondences on average than the regular-grid and stochastic controls. In particular, the Sierpinski pattern produces a larger pool of geometrically consistent correspondences than the smoother circle-fractal baseline, consistent with its dense multi-scale edges and corners. Based on this evidence, we adopt the Sierpinski triangle as the default dome texture. It supplies dense, geometrically verifiable cues in dome-conditioned regions, providing additional camera-motion evidence beyond the warped RGB signal.

5.4 Ablation Study

Dome texture. We first compare five dome textures through controlled end-to-end ablations on Charge: solid-black filling, checkerboard, Perlin noise, random dots, and Sierpinski. Whereas Sec. 5.3 analyzes pattern trackability in isolation, this end-to-end ablation directly measures how the choice of dome texture affects the trained video-generation model. As shown in Tab. 3-(a), Sierpinski achieves the strongest geometric consistency and target-view fidelity among the tested textures, while also providing the best background consistency and rotational accuracy. Random Dots yields a lower ATE, and solid-black filling slightly better preserves subject consistency, but neither provides the same overall performance across geometry, camera control, and fidelity. These results show that the benefit does not arise merely from filling disoccluded regions with an arbitrary dense texture.

RoPE indexing strategies. We next examine how the RoPE assignment of source-video tokens affects generation under the same Charge protocol. We compare NegRoPE with Frame Concat

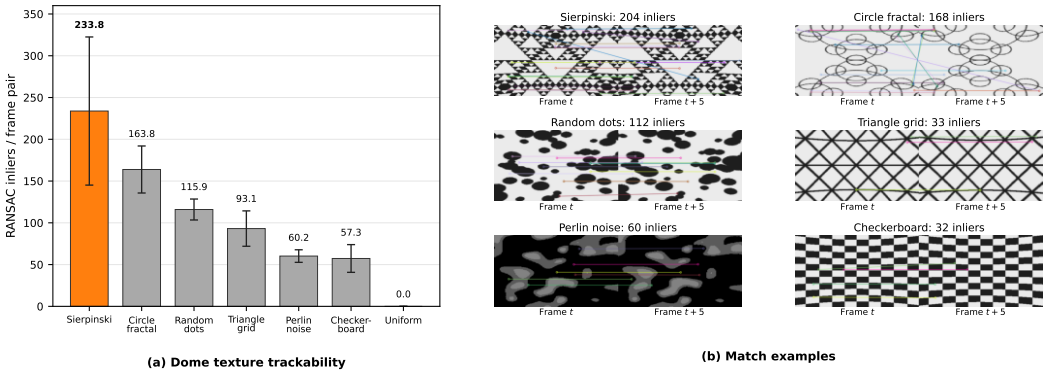

Figure 5: **(a)** Pattern-only trackability over 10 camera trajectories. Ours (Sierpinski) yields the most RANSAC-verified SIFT inliers per frame pair. **(b)** Representative frame-t to frame-$t{+}5$ matches; for clarity, only the top 10% inliers by Lowe ratio are drawn, while titles report total inliers.

Table 3: **Ablations on Charge. (a)** The Sierpinski dome provides the strongest geometric consistency and target-view fidelity among the tested textures, whereas Random Dots yields the lowest ATE. **(b)** NegRoPE best preserves content and 3D consistency; removing source spatial RoPE improves pose accuracy at a substantial cost to content consistency and fidelity. All variants are independently trained under matched settings. **Bold**: best; underline: second best within each ablation.

	Variant	Visual Quality		Geometric Consistency		Camera Accuracy		Fidelity	
		Subject Cons.↑	Background Cons.↑	Dyn-MEt3R↑	MEt3R↓	RotErr↓	ATE↓	PSNR↑	LPIPS↓
(a) Dome	Solid Black (No Texture)	**0.9251**	0.9504	0.8218	0.5622	12.2411	0.2982	17.2483	0.4455
	Checkerboard	0.9195	0.9462	0.8273	0.5773	11.7870	0.2821	17.1045	0.4505
	Perlin Noise	0.9188	0.9471	0.8112	0.5741	12.0491	0.3000	17.0997	0.4514
	Random Dots	0.9215	0.9484	0.8201	0.5745	11.5109	**0.2769**	17.3298	0.4492
	Sierpinski (Ours)	0.9208	**0.9566**	**0.8310**	**0.3492**	**11.2265**	0.2951	**17.9980**	**0.4433**
(b) RoPE	Frame Concat	0.8923	0.9471	0.7590	0.3916	12.7208	0.2991	17.1581	0.5305
	No RoPE Mod.	0.8168	0.9268	0.6726	0.6683	11.3965	0.2918	10.7398	0.7064
	Offset RoPE (+4096)	0.8118	0.9206	0.7278	0.5666	13.1224	0.3455	16.6975	0.5903
	No Spatial RoPE	0.8880	0.9322	0.7660	0.7548	**11.1133**	**0.2665**	13.6331	0.6782
	Temporal Offset (+0.1)	0.8921	0.9466	0.7783	0.3883	11.2191	0.3062	17.2652	0.5377
	NegRoPE (Ours)	**0.9208**	**0.9566**	**0.8310**	**0.3492**	11.2265	0.2951	**17.9980**	**0.4433**

and four alternative indexing schemes: shared source–target RoPE indices, a large positive spatial offset, removal of source spatial RoPE, and a small temporal offset. Except for Frame Concat, these variants use the same source-token concatenation and differ only in their RoPE index assignment, directly isolating the effect of NegRoPE. As shown in Tab. 3(b), NegRoPE achieves the strongest content consistency, geometric consistency, and target-view fidelity among the evaluated strategies. Removing source spatial RoPE yields lower camera-pose errors, but substantially degrades content preservation, geometric consistency, and fidelity. We observe cases in which this variant generates subjects with altered identities or backgrounds inconsistent with the source video, suggesting that it uses source-video information less reliably. NegRoPE avoids this trade-off by retaining spatial cues within the source stream while assigning source and target tokens to disjoint spatial RoPE domains.

5.5 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 [10], 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. 4 suggest

Table 4: **ReAngle-A-Video extension.** The Sierpinski cue improves translation accuracy and ATE.

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

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.

7 Acknowledgement

This work was supported in part by the National Science Foundation under CAREER Award No. 2544116. We also acknowledge scholarship support from SCB Public Company Limited and PTT Public Company Limited (Thailand).

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] Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, et al. Vipe: Video pose engine for 3d geometric perception. *arXiv preprint arXiv:2508.10934*, 2025.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10486–10496. IEEE, 2025.

[14] 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.

[15] 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.

[16] 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.

[17] Michal Nazarczuk, Thomas Tanay, Arthur Moreau, Zhensong Zhang, and Eduardo Pérez-Pellitero. Charge: A comprehensive novel view synthesis benchmark and dataset to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15323–15333, 2026.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[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

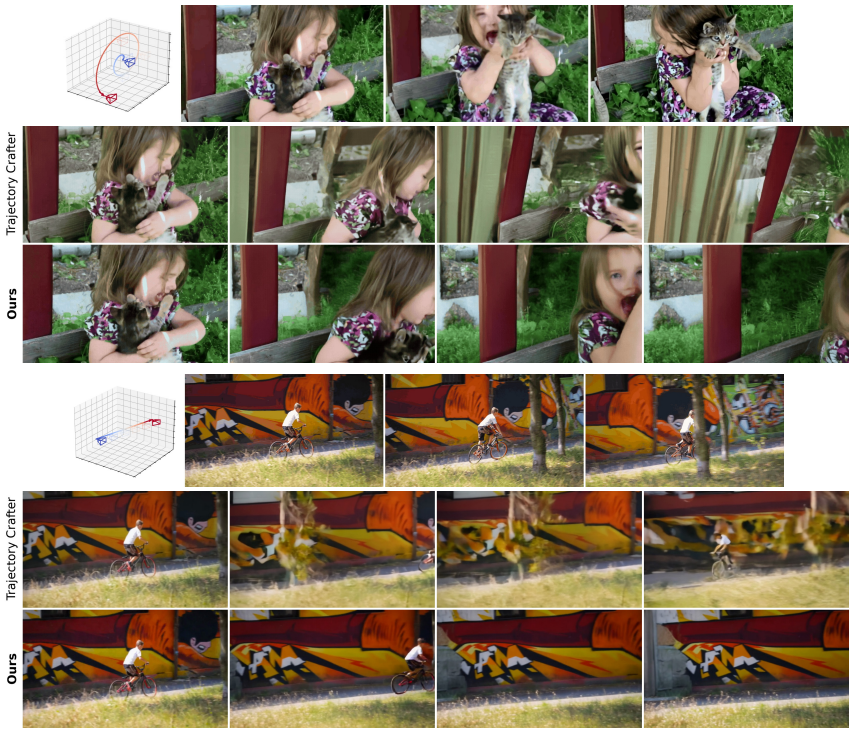

Figure 6: **Failure cases of TrajectoryCrafter.** When the target camera moves beyond the original scene coverage, it may hallucinate content or fail to follow the camera pose, especially as the main object becomes small or leaves the target-view frustum.

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.

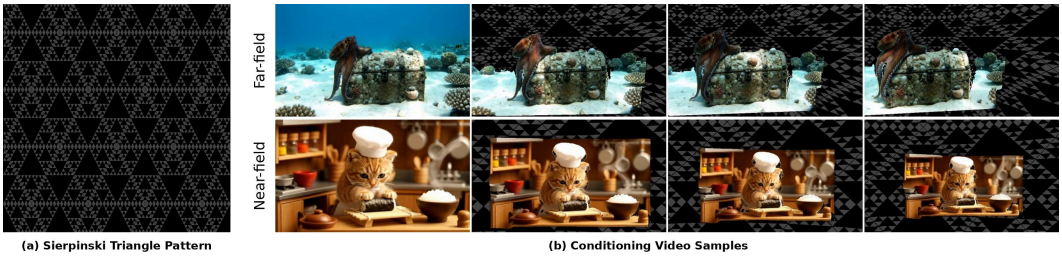

Figure 7: **Geometry proxy design and conditioning video examples. (a)** The Sierpinski triangle pattern used as a geometry proxy for camera motion conditioning. Its self-similar structure provides rich spatial cues without scene-specific texture, enabling generalizable control. **(b)** Example conditioning 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[3], 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 controllability by estimating the camera trajectory of each generated video and comparing it with the reference trajectory. To select the pose estimator, we benchmark DA3 [14], ViPE [8], and MegaSaM [13] on Charge, which provides ground-truth camera trajectories.

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

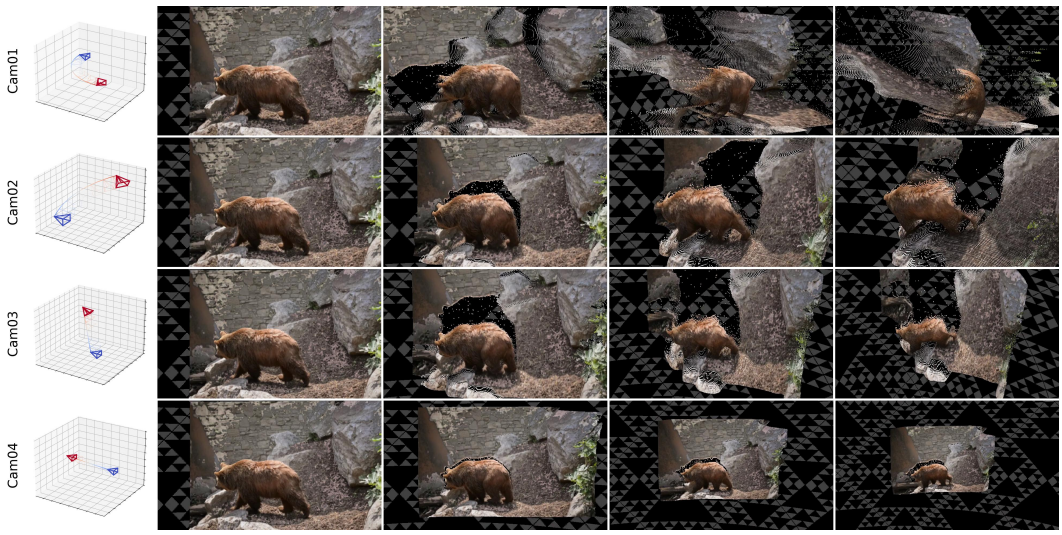

Figure 8: **Additional camera trajectories used for evaluation.** Each row shows the geometry proxy frames used as video conditioning input for four additional camera trajectories. The sampled frames illustrate the spatial extent and viewpoint variation induced by each trajectory.

As shown in Table 5, ViPE achieves the lowest mean RotErr and ATE and ties for the lowest mean TransErr. We therefore use ViPE throughout our evaluation. This choice also avoids coupling the evaluation to DA3, which is used to construct our conditioning videos.

Table 5: Pose estimator validation on Charge. **Bold**: best; underline: second best.

Estimator	RotErr↓ Mean (Median)	TransErr↓ Mean (Median)	ATE↓ Mean (RMSE)
DA3	0.6442 (0.4051)	0.1961 (0.1368)	0.1751 (0.2025)
ViPE	**0.6151** (0.4184)	**0.1919** (0.1292)	**0.1691** (0.1966)
MegaSAM	0.6283 (0.4043)	**0.1919** (0.1521)	0.1717 (0.1937)

C.4 User Study

For user study, participants were recruited through Prolific[4] 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 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 with selected baselines beyond the main evaluation dataset. These results further illustrate the generalization of SierpinskiCam across different

[4]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 6: Instructions used for the user study.

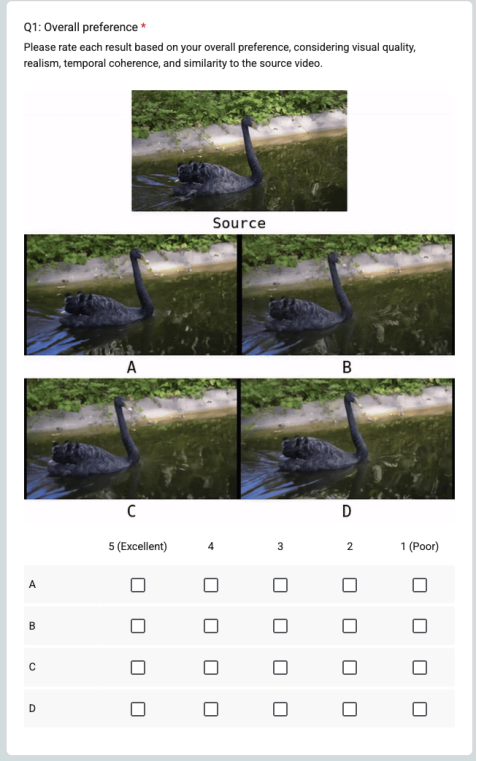

Figure 9: Screenshot of the user study interface.

data sources and camera-motion scenarios. We also include supplementary videos to better visualize camera-control behavior and temporal dynamics.

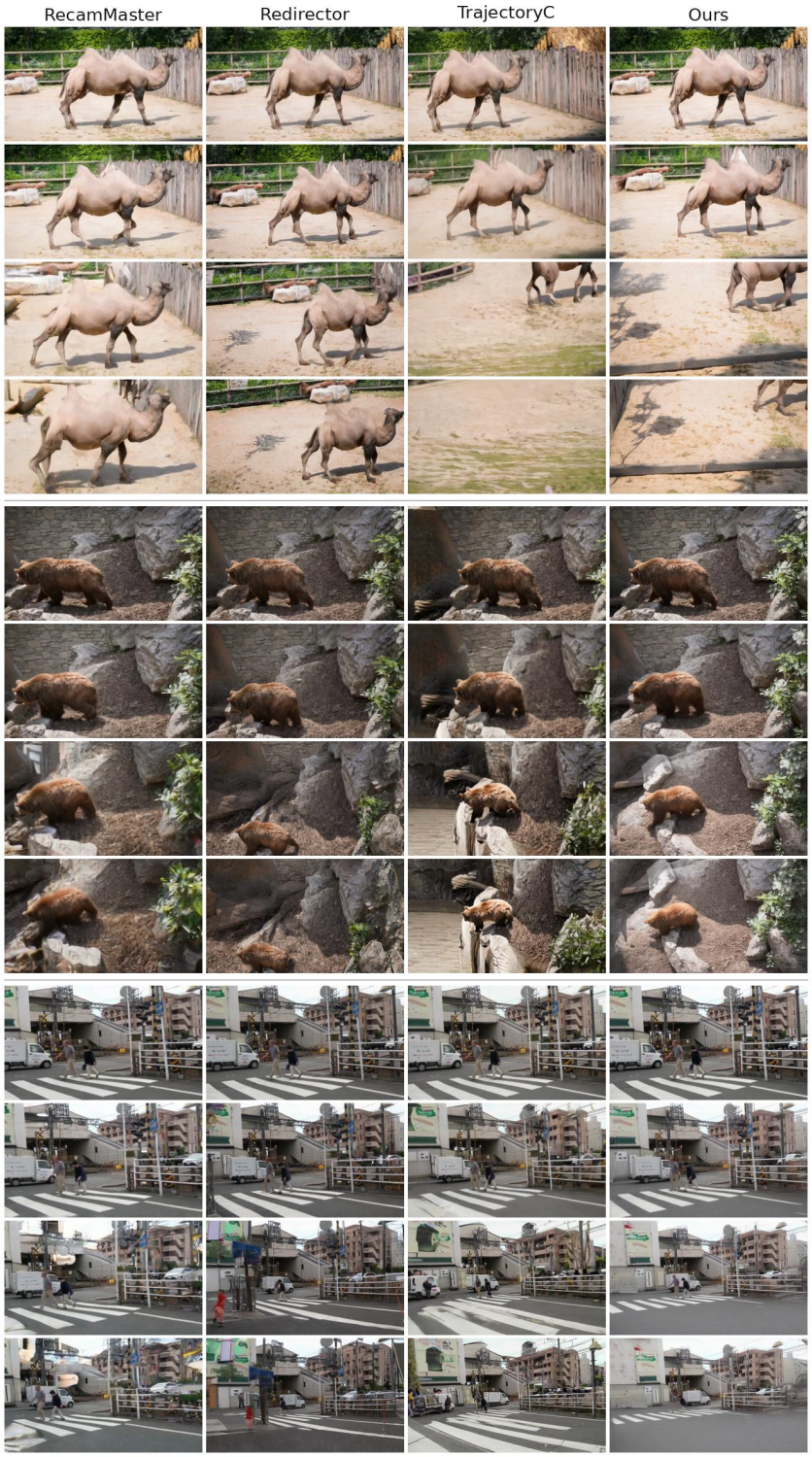

Figure 10: **Additional comparison on newly added challenging camera trajectories.**



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

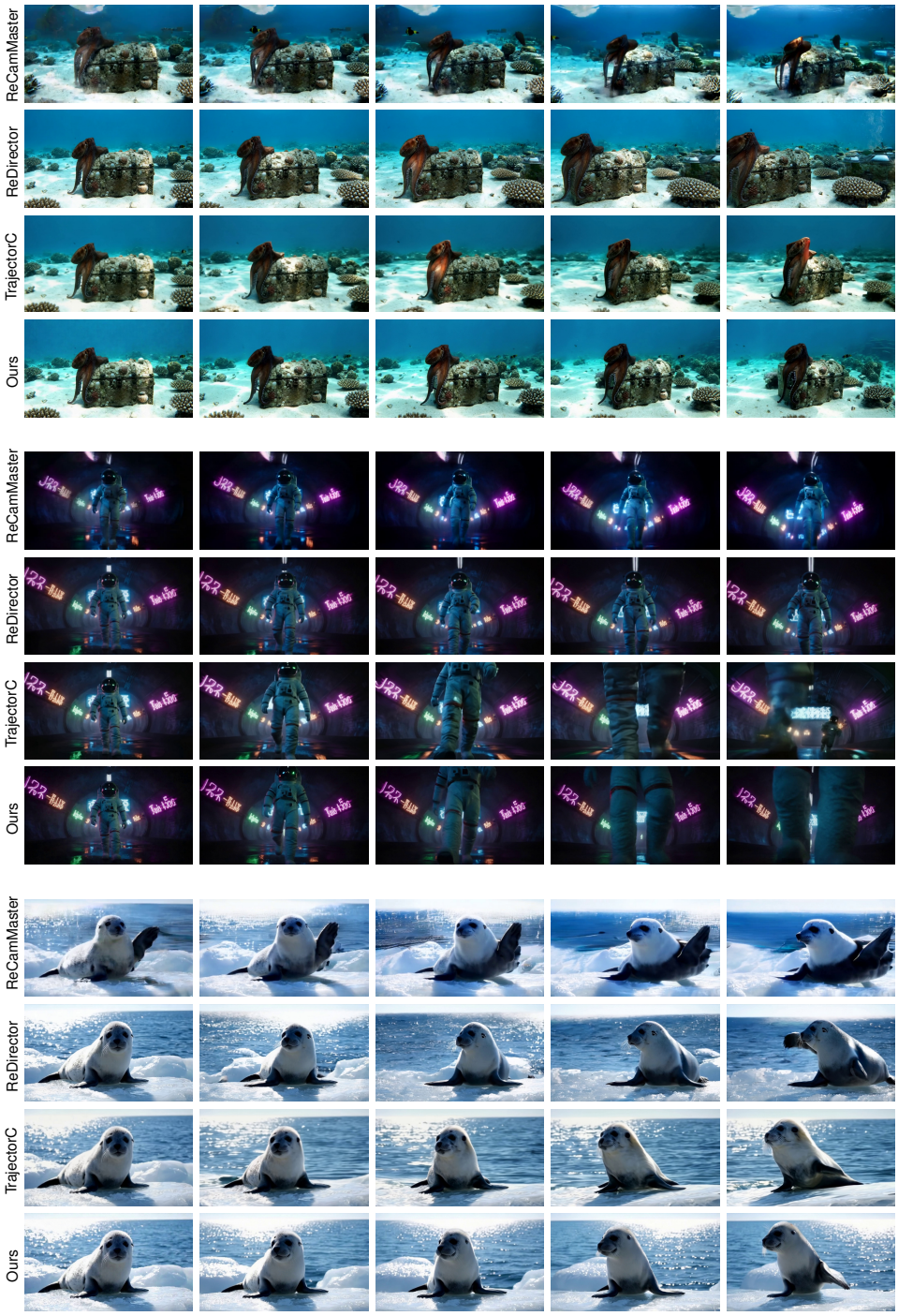

Figure 12: **Additional qualitative comparison on generated video by Veo**.

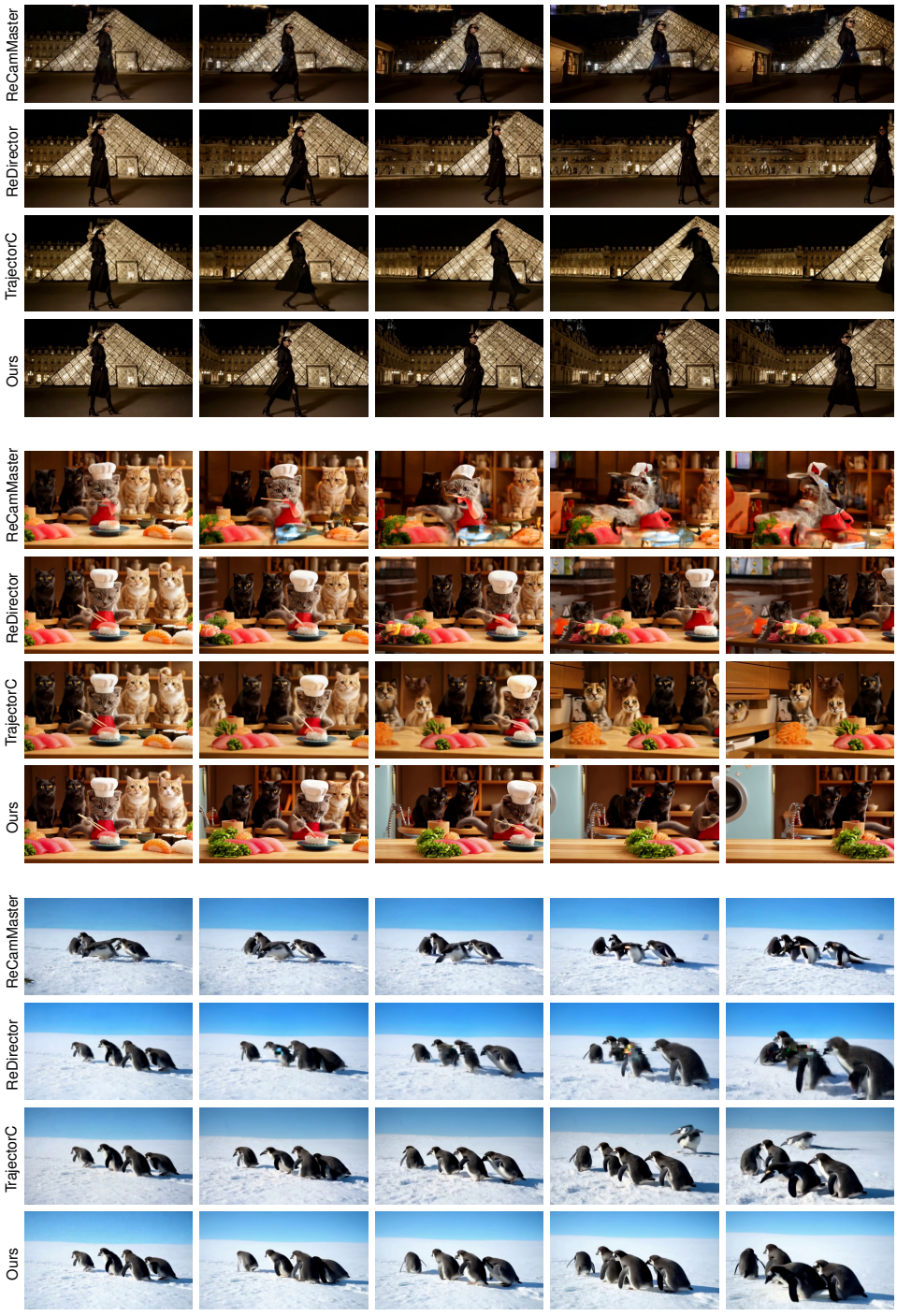

Figure 13: **Additional qualitative comparison on generated video by Veo**.