CORRRETIME: CORRESPONDENCE-GUIDED OBJECT-WISE SPACE-TIME RETIMING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent video diffusion methods have explored temporal control, but typically apply a single global transformation to the entire scene, making it difficult to independently control the temporal evolution of multiple objects and the camera/background within the same video. We introduce CorrRetime, a controllable video diffusion framework for fine-grained object-wise temporal remapping with independent camera/background retiming. Our key idea is to represent this control through explicit spatiotemporal correspondences that specify, for each output frame, both the source-time state of each object and its target location under the retimed camera trajectory. This enables multiple foreground objects to independently accelerate, decelerate, pause, resume, or reverse while preserving their underlying source motion, while the camera/background can be independently retimed along the original camera path according to a separate temporal schedule. To enable supervised training and systematic evaluation of this task, we construct a large-scale synthetic dataset with independently parameterized object and camera temporal schedules, spanning diverse multi-object retiming patterns, camera trajectories, and occlusion relationships. We evaluate CorrRetime on both synthetic and real-world videos, demonstrating accurate object-wise and camera/background temporal control while outperforming adapted baselines in retiming accuracy and visual quality.

1 INTRODUCTION

Videos capture dynamic scenes that evolve over time, with each object exhibiting its own motion and state changes. However, all object-level dynamics in a captured video are constrained to a single shared timeline. The ability to independently control the temporal progression of individual objects is important for fine-grained video editing and generation, enabling diverse temporal manipulations such as object-wise slow motion, acceleration, pausing, and reversal.

Recent advances in video diffusion models have enabled photorealistic generation with strong modeling of motion and temporal dynamics (Wan Team et al., 2025; Kong et al., 2024; Yang et al., 2025). Recent temporal control methods further support scene-level manipulations (Huang et al., 2026; Wang et al., 2026). However, these temporal-control approaches generally apply a shared temporal mapping to the whole scene and cannot directly support independent temporal transformations for multiple objects. Meanwhile, trajectory- or motion-conditioned methods can specify *where* individual objects should move (Wang et al., 2025a; Wu et al., 2024; Shi et al., 2024), but not *which temporal state* should appear at a given output frame. Thus, they are insufficient for independently remapping object timelines while preserving the original motion and appearance.

Motivated by this gap, we study object-wise temporal remapping. Given a dynamic-scene video, our goal is to independently remap the temporal progression of each object while preserving the original scene appearance and underlying motion content. We additionally allow the camera/background track to follow a separate temporal schedule along the original camera trajectory. Representative results under different object-wise and camera retiming schedules are shown in Fig. 1.

Object-wise temporal remapping poses unique challenges because changing an object's temporal progression places its state at a different spatiotemporal location than in the source video. Accurate control therefore requires specifying not only *when* an object state originates, but also *where* it should appear in the target video. Independently retiming multiple objects also changes occlusion

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: **Object-wise space-time retiming with CorrRetime.** Given a single source video, CorrRetime independently controls the temporal progression of individual foreground objects and the camera/background. Different retiming schedules can produce distinct temporal evolutions from the same input, including non-uniform speed changes and temporal reversal.

relationships, often revealing regions unseen in the source. Simple warping or interpolation cannot synthesize such missing content, while temporal parameters alone (e.g., speed or timestamps) provide insufficient spatial guidance. Object-wise temporal remapping therefore requires a representation that explicitly links source object states to target spatiotemporal locations while retaining the ability to synthesize newly revealed regions.

To address these challenges, we introduce CorrRetime, a correspondence-guided video diffusion framework for object-wise temporal remapping. Our key idea is to explicitly construct spatiotemporal correspondences between object states in the source video and their desired spatiotemporal locations in the retimed target video. Each correspondence specifies *when* in the source timeline the object state occurs and *where* it should appear in the target video, thereby providing explicit guidance for both temporal progression and spatial placement.

We incorporate these correspondences into a pretrained video diffusion model (Wan Team et al., 2025), enabling it to preserve object appearance and motion while handling occlusions and synthesizing newly revealed regions. Unlike global temporal control methods, CorrRetime allows multiple objects in the same scene to follow independent temporal schedules, supporting acceleration, deceleration, stopping, reversing, and piecewise-varying transformations while maintaining visual and temporal coherence. Extensive experiments demonstrate that CorrRetime accurately follows object-specific temporal controls while preserving visual quality and scene consistency.

Training and evaluating object-wise space-time retiming requires videos with independently controllable object and camera timelines and accurate correspondences, which are difficult to obtain from real-world videos. We therefore construct a large-scale synthetic dataset with independently parameterized object and camera schedules, providing paired supervision and ground-truth retimed videos for supervised training and quantitative evaluation.

In this work, our contributions can be summarized as: (i) we formulate object-wise space-time retiming, where multiple foreground objects and the camera/background timeline can be independently retimed while preserving the original scene appearance and motion content; (ii) we introduce CorrRetime, a correspondence-guided video diffusion framework that links each object's source-time state to its target location under the retimed camera trajectory; and (iii) we construct a large-scale synthetic dataset with paired supervision for independently parameterized object and camera temporal schedules, enabling supervised training and quantitative evaluation of this task.

2 Related Work

Video Generative Models. Diffusion models have been extended from image generation to video synthesis by jointly modeling spatial appearance and temporal dynamics (Ho et al., 2022b;a; Singer et al., 2023). Early approaches introduced temporal modules into image diffusion architectures or operated in compressed latent spaces (Blattmann et al., 2023b;a; Guo et al., 2024; Chen et al., 2023; Xing et al., 2024), while subsequent works improved motion dynamics and generation quality through stronger spatiotemporal modeling (Bar-Tal et al., 2024). More recently, transformer-based models such as CogVideoX (Yang et al., 2025), HunyuanVideo (Kong et al., 2024), Wan (Wan Team

et al., 2025), and Movie Gen (Polyak et al., 2024) have provided strong generative priors for video generation.

Temporal Control and Video Retiming. Temporal video manipulation has been studied through video retiming and motion-aware editing. Earlier work such as Layered Neural Rendering (Lu et al., 2020) enables per-person retiming via independently editable layers. Recent methods provide generative temporal control: SpaceTimePilot (Huang et al., 2026) and BulletTime (Wang et al., 2026) support flexible scene-level time manipulation, while 4DiM (Watson et al., 2025), CAT4D (Wu et al., 2025), and Lyra (Bahmani et al., 2026) jointly model space and time. TrajectoryCrafter (Yu et al., 2025) and ReCamMaster (Bai et al., 2025) focus on camera-controlled video re-rendering. In contrast, CorrRetime focuses on temporal remapping while preserving the source motion content, explicitly assigning each object its own source-time mapping and independently retiming the camera/background timeline through spatiotemporal correspondences.

Motion-Controlled Video Generation. Recent works introduce explicit motion representations for fine-grained video control, conditioning on camera parameters, object trajectories, or bounding boxes (Wang et al., 2024; Ma et al., 2024; He et al., 2025). Point trajectories have become a flexible representation for object-level control (Geng et al., 2025; Shi et al., 2024; Wu et al., 2024; Yin et al., 2023; Zhang et al., 2025; Wang et al., 2025a; Lee et al., 2026). Other methods use alternative motion controls, such as masks and bounding boxes (Li et al., 2025a), or camera-motion trajectories (Xiao et al., 2025). 3D-aware methods such as Diffusion as Shader (Gu et al., 2025) and LeviTor (Wang et al., 2025b) further incorporate geometric or depth-aware guidance. MotionV2V (Burgert et al., 2026) further extends track-based control to object-level video motion editing by manipulating sparse point tracks.

These methods effectively control *where* objects move, but typically do not explicitly specify *which source-time state* should occupy each target location. This distinction is crucial for object-wise retiming, which preserves object dynamics while altering their temporal progression. Our method addresses this through explicit source-to-target spatiotemporal correspondences.

3 METHODS

3.1 OVERVIEW

In this paper, we propose CorrRetime, a video-to-video framework for object-wise temporal remapping. Given a source video $V^{\mathrm{src}} \in \mathbb{R}^{F\times C\times H\times W}$, we generate a target video $V^{\mathrm{tgt}} \in \mathbb{R}^{F\times C\times H\times W}$, where multiple foreground objects are independently retimed according to their own temporal schedules and may therefore originate from different source moments at the same target frame.

The camera trajectory can be retimed independently of the foreground objects, while the background follows the same temporal schedule as the camera. In other words, the camera/background timeline is independent of the foreground object timelines, so they may correspond to different moments in the source video at the same target frame.

Retiming an object or the camera requires generating its appearance at a new spatiotemporal location rather than simply copying it from the source. The challenge becomes greater when an object's source time differs from the camera's, since the resulting combination of object and camera states was never jointly observed in the source video. Object-wise temporal remapping under independent camera retiming is therefore inherently spatiotemporal: the model must determine both *when* each state should be drawn from and *where* each object should appear in the resulting view.

Our key idea is to represent this control explicitly through source-to-target spatiotemporal correspondences, rather than relying solely on scalar time or speed values. For each object, we link its state in the source video to its desired location in the target video under the retimed camera trajectory. We leverage the 3D geometry between the source and target views to determine where each object state should be reprojected. These correspondences, together with source-video context and background guidance, condition a pretrained video diffusion model to preserve scene structure while handling occlusions, disocclusions, and newly revealed regions. An overview of the CorrRetime framework is shown in Fig. 2.

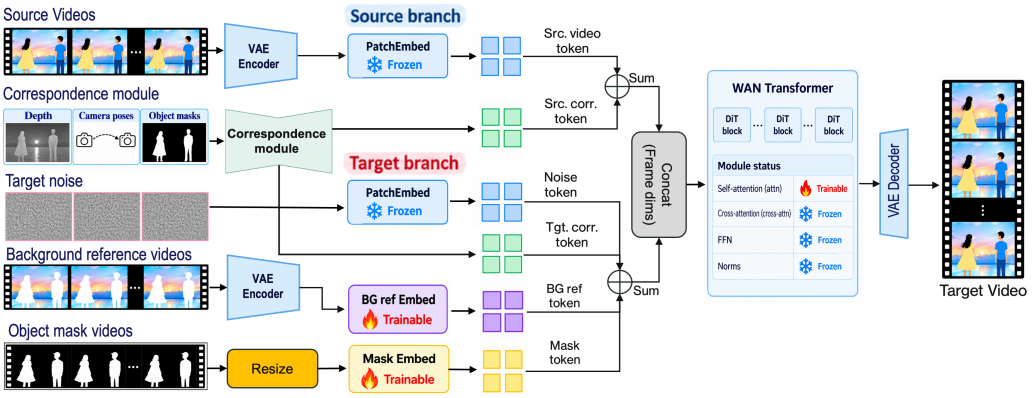

Figure 2: **Overview of CorrRetime.** Correspondence tokens are added to both source and target latent tokens, while the target branch additionally receives background-reference and object-mask conditioning tokens. The two branches are then concatenated and jointly processed by the Wan transformer. During training, we fine-tune the self-attention layers and newly introduced conditioning modules, while keeping the remaining pretrained components frozen.

3.2 Correspondence-Guided Video Diffusion Model

Given the temporal mapping $\tau_k(f)$ for each object, we construct explicit source-to-target correspondences that connect each object state in the source video to its desired location in the target video, and use them as geometric conditioning for a pretrained video diffusion model.

Per-Object Spatiotemporal Correspondence Construction. To enable object-wise retiming, the user specifies an independent playback-speed schedule for each object, allowing it to move faster, slower, pause, or reverse. Based on this schedule, for each object and target frame, we determine the corresponding source-time state and construct explicit correspondences to the target location using per-object segmentation masks, depth, and camera poses—lifting source pixels to 3D, transforming them into the target camera frame, and projecting them onto the target image plane.

We formalize this via an object-specific temporal mapping $\tau_k(f)$, specifying the source time from which object O_k's state is taken at target frame f. Given the playback-speed schedule $s_k(f)$, this mapping is obtained by accumulating the speed over time:

$$\tau_k(f) = \tau_k(0) + \sum_{n=0}^{f-1} s_k(n). \tag{1}$$

The corresponding source frame index is obtained by rounding and clamping $\tau_k(f)$ to the valid frame range,

$$t_{k,f}^{\mathrm{src}} = \mathrm{clip}\left(\mathrm{round}(\tau_k(f)); [0, F-1]\right), \tag{2}$$

so that $\tau_k(f)$ itself may momentarily exceed $[0, F-1]$, while $t_{k,f}^{\mathrm{src}}$ always indexes a valid source frame. Varying $s_k(f)$ over time naturally supports acceleration ($s_k > 1$), deceleration ($0 < s_k < 1$), intentional pausing ($s_k = 0$), and temporal reversal ($s_k < 0$); in addition, once $\tau_k(f)$ drifts past a boundary, the clamped index saturates there, effectively holding the object in a paused state until $\tau_k(f)$ re-enters the valid range.

Similarly, we independently retime the camera trajectory using a camera playback speed $s_{\mathrm{cam}}(f)$. Using the same accumulation, we obtain the corresponding camera time $\tau_{\mathrm{cam}}(f) = \tau_{\mathrm{cam}}(0) + \sum_{n=0}^{f-1} s_{\mathrm{cam}}(n)$, and from this obtain the target camera pose as $\mathbf{P}^{\mathrm{tgt}}(f) = \mathbf{P}^{\mathrm{src}}\big(\tau_{\mathrm{cam}}(f)\big)$.

We sample points within the object's segmentation mask at source frame $t_{k,f}^{\mathrm{src}}$ and lift them to 3D using their corresponding depth, obtaining $\mathbf{X}_{i,\mathrm{cam}}^{\mathrm{src}}$ for each point i. Let $\mathbf{P}(t) \in SE(3)$ denote the camera-to-world transformation at time t, so that $\mathbf{P}^{\mathrm{src}}(t)$ and $\mathbf{P}^{\mathrm{tgt}}(f)$ give the poses of the source and (retimed) target cameras, respectively. Writing $\tilde{\mathbf{X}} = [\mathbf{X}^\top, 1]^\top$ for the homogeneous coordinate of a point $\mathbf{X}$, each point is transformed into the target camera's coordinate frame as

$$\tilde{\mathbf{X}}_{i,\mathrm{cam}}^{\mathrm{tgt}} = \big(\mathbf{P}^{\mathrm{tgt}}(f)\big)^{-1} \mathbf{P}^{\mathrm{src}}\big(t_{k,f}^{\mathrm{src}}\big)\, \tilde{\mathbf{X}}_{i,\mathrm{cam}}^{\mathrm{src}}, \tag{3}$$

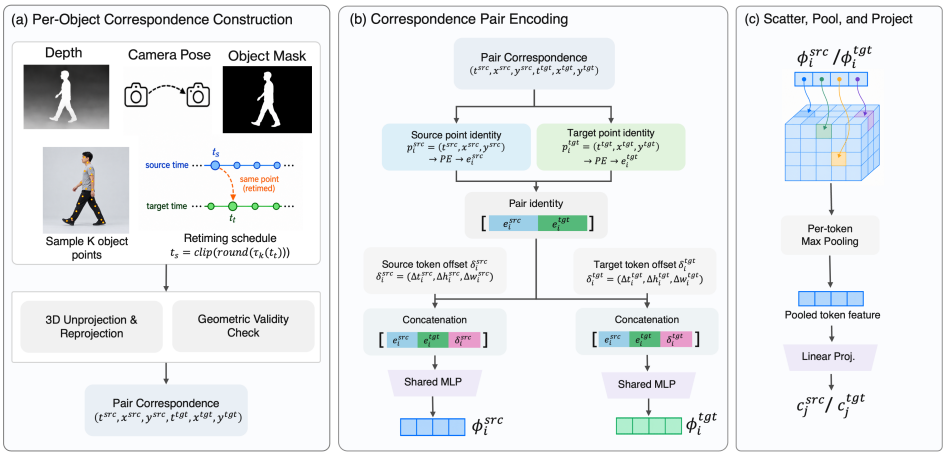

Figure 3: Overview of the correspondence module: (a) per-object correspondence construction from temporal mappings and 3D geometry, (b) pair encoding of source/target coordinates and within-token offsets, and (c) token-wise aggregation.

which maps a point expressed in the source camera's coordinate frame at time $t^{\mathrm{src}}_{k,f}$ into the coordinate frame of the target camera at frame f. Since $t^{\mathrm{src}}_{k,f}$ depends on the object's own playback speed s_k, while $\tau_{\mathrm{cam}}(f)$ depends on the independently retimed camera speed s_{cam}, the source and target poses are generally evaluated at different, independently varying times; as a result, this relative transformation varies not only across target frames f but also across objects k.

Together with $t^{\mathrm{src}}_{k,f}$ and $t^{\mathrm{tgt}} = f$, the projected target location $(x^{\mathrm{tgt}}_i, y^{\mathrm{tgt}}_i)$ specifies *which* object state should be transferred, *when* it should appear, and *where* it should appear in the target video. We apply visibility filtering to discard invalid projections and resolve multiple points projecting to the same target location, followed by stratified spatial sampling to obtain a fixed-size, spatially well-distributed set of correspondences.

Fixing O_k and f, we denote each correspondence as a pair $(\mathbf{p}^{\mathrm{src}}_i, \mathbf{p}^{\mathrm{tgt}}_i)$ with $\mathbf{p}^{\mathrm{src}}_i = (t^{\mathrm{src}}, x^{\mathrm{src}}_i, y^{\mathrm{src}}_i)$ and $\mathbf{p}^{\mathrm{tgt}}_i = (t^{\mathrm{tgt}}, x^{\mathrm{tgt}}_i, y^{\mathrm{tgt}}_i)$. Collecting all such correspondences yields the object control signal

$$\mathcal{C}_k(f) = \left\{ \left(\mathbf{p}^{\mathrm{src}}_i, \mathbf{p}^{\mathrm{tgt}}_i \right) \right\}_{i=1}^{N_{k,f}} . \tag{4}$$

We inject the correspondence signal into a pretrained video DiT using a pairwise spatiotemporal encoding of the source and target coordinates. The encoding is added to the corresponding latent tokens, providing explicit geometric guidance for object retiming.

Pairwise correspondence conditioning To condition the video diffusion model on the constructed correspondences, we convert each correspondence pair $(\mathbf{p}^{\mathrm{src}}_i, \mathbf{p}^{\mathrm{tgt}}_i)$ into a feature representation aligned with the latent token space.

For each valid correspondence, we normalize the temporal and spatial components of the source and target coordinates by their respective coordinate ranges, yielding normalized spatiotemporal coordinates $\mathbf{u}^{\mathrm{src}}_i, \mathbf{u}^{\mathrm{tgt}}_i \in [0,1]^3$. We then apply a sinusoidal positional encoding (SinPE) to the normalized source and target coordinates separately,

$$\mathbf{e}^{\mathrm{src}}_i = \mathrm{sinPE}(\mathbf{u}^{\mathrm{src}}_i), \qquad \mathbf{e}^{\mathrm{tgt}}_i = \mathrm{sinPE}(\mathbf{u}^{\mathrm{tgt}}_i). \tag{5}$$

To preserve the association between endpoints, we concatenate their positional embeddings in a fixed order, $[\mathbf{e}^{\mathrm{src}}_i | \mathbf{e}^{\mathrm{tgt}}_i]$, forming a shared correspondence identity that explicitly links the source and target points, as illustrated in Fig. 3(b).

Because the latent token grid has much lower spatiotemporal resolution than the original pixel space, multiple pixel locations map to the same latent token. To retain precise localization, we decompose each correspondence endpoint into a discrete token index and a continuous within-token offset.

For a point at pixel coordinate (t, h, w), we map its spatial components (h, w) to token-grid units, where the integer parts determine the assigned token and the fractional parts specify its within-

token position. We rescale the fractional parts to $[-1, 1]$ and use them as spatial offsets $(\Delta h, \Delta w)$, preserving fine-grained locations within the same token.

Along the temporal axis, multiple raw frames are compressed into a single latent token. In our video VAE, the first latent token corresponds to frame 0, while each subsequent token represents four consecutive frames. We therefore encode the relative frame position within each temporal token using Δt, mapping the four positions uniformly to $[-1, 1]$ as $(-1, -\frac{1}{3}, \frac{1}{3}, 1)$.

We define the source and target offsets for correspondence i as $\boldsymbol{\delta}_i^{\mathrm{src}} = (\Delta t_i^{\mathrm{src}}, \Delta h_i^{\mathrm{src}}, \Delta w_i^{\mathrm{src}})$ and $\boldsymbol{\delta}_i^{\mathrm{tgt}} = (\Delta t_i^{\mathrm{tgt}}, \Delta h_i^{\mathrm{tgt}}, \Delta w_i^{\mathrm{tgt}})$, computed separately for the two endpoints. The token indices determine where the correspondence features are scattered, while the offsets preserve their fine-grained within-token positions.

$$\boldsymbol{\phi}_i^{\mathrm{src}} = g_\theta\big([\mathbf{e}_i^{\mathrm{src}} \,|\, \mathbf{e}_i^{\mathrm{tgt}} \,|\, \boldsymbol{\delta}_i^{\mathrm{src}}]\big), \qquad \boldsymbol{\phi}_i^{\mathrm{tgt}} = g_\theta\big([\mathbf{e}_i^{\mathrm{src}} \,|\, \mathbf{e}_i^{\mathrm{tgt}} \,|\, \boldsymbol{\delta}_i^{\mathrm{tgt}}]\big). \tag{6}$$

We use the same encoder g_θ for both endpoints, sharing the correspondence representation while retaining endpoint-specific offsets. The resulting features are scattered to the source and target latent tokens indexed by their discretized endpoint coordinates, respectively.

As illustrated in Fig. 3(c), multiple correspondence points can map to the same latent token. We therefore aggregate their features via element-wise max pooling independently for the source and target sides. The pooled features are then projected to the transformer's hidden dimension using a shared linear layer W_{proj}. Let $\mathcal{P}^{\mathrm{src}}(j)$ and $\mathcal{P}^{\mathrm{tgt}}(j)$ denote the source- and target-side point features assigned to latent token j. The resulting token-level correspondence features are

$$\mathbf{c}_j^{\mathrm{src}} = W_{\mathrm{proj}} \max_{\boldsymbol{\phi}^{\mathrm{src}} \in \mathcal{P}^{\mathrm{src}}(j)} \boldsymbol{\phi}^{\mathrm{src}}, \qquad \mathbf{c}_j^{\mathrm{tgt}} = W_{\mathrm{proj}} \max_{\boldsymbol{\phi}^{\mathrm{tgt}} \in \mathcal{P}^{\mathrm{tgt}}(j)} \boldsymbol{\phi}^{\mathrm{tgt}}, \tag{7}$$

where $\max$ is taken element-wise over the feature dimension, and tokens with no assigned points remain all-zero. This yields correspondence feature grids, $\mathbf{c}^{\mathrm{src}}$ and $\mathbf{c}^{\mathrm{tgt}}$, aligned with the source and target latent token grids and added to the corresponding patch-embedded latent inputs.

Camera and background retiming conditioning. While foreground objects are independently retimed by their temporal mappings $\tau_k(f)$, the background follows the camera mapping $\tau_{\mathrm{cam}}(f)$. Since camera retiming only changes temporal progression along the original camera path, we use the source frame corresponding to $\tau_{\mathrm{cam}}(f)$ as the background reference for target frame f.

For each target frame, we remove all controllable foreground objects from the reference using their segmentation masks and fill the erased regions with a constant value. The resulting foreground-erased frame is encoded into a VAE latent $\mathbf{z}_{\mathrm{bg}}$. We also provide a binary mask $\mathbf{m}$, formed by the union of the erased object masks, to indicate regions that should be synthesized by the model.

Thus, $\mathbf{z}_{\mathrm{bg}}$ provides dense appearance guidance for the background, while $\mathbf{m}$ marks regions where the retimed foreground objects should be synthesized. We embed $\mathbf{z}_{\mathrm{bg}}$ and $\mathbf{m}$ using separate 3D convolutional layers, E_{bg} and E_{mask}, respectively, with kernel sizes and strides matching the video patch embedding to align their features with the latent token grid.

Conditioning Injection. We inject all conditioning signals at the patch embedding stage, before the resulting tokens enter the transformer.

Source tokens. The clean source video is patch-embedded and combined with the source-side correspondence features $\mathbf{c}^{\mathrm{src}}$,

$$\mathbf{h}^{\mathrm{src}} = \mathrm{PatchEmbed}(\mathbf{x}^{\mathrm{src}}) + \mathbf{c}^{\mathrm{src}}, \tag{8}$$

where $\mathbf{x}^{\mathrm{src}}$ is the clean source video latent.

Target tokens. The noisy target video is patch-embedded and combined with the target-side correspondence features together with the background and mask embeddings introduced above,

$$\mathbf{h}^{\mathrm{tgt}} = \mathrm{PatchEmbed}(\mathbf{x}_{\mathrm{noisy}}^{\mathrm{tgt}}) + \mathbf{c}^{\mathrm{tgt}} + E_{\mathrm{bg}}(\mathbf{z}_{\mathrm{bg}}) + E_{\mathrm{mask}}(\mathbf{m}), \tag{9}$$

where $\mathbf{x}_{\mathrm{noisy}}^{\mathrm{tgt}}$ is the noised target video latent at the current diffusion timestep. Here, $\mathbf{c}^{\mathrm{tgt}}$ specifies where the retimed object should appear, $E_{\mathrm{bg}}(\mathbf{z}_{\mathrm{bg}})$ provides the surrounding background appearance, and $E_{\mathrm{mask}}(\mathbf{m})$ identifies regions where foreground objects were removed from the background reference.



Figure 4: Qualitative results on real-world videos under independent object-wise and camera retiming. For each example, the top row shows the source video and the bottom row shows the CorrRetime output. Colored markers denote controlled foreground objects, and accompanying values indicate their playback speeds. The value next to the camera icon denotes camera playback speed. The anchor frame, where object and camera timelines coincide with the source video, is highlighted in red.

Transformer input. We then form the DiT input by concatenating the source and target tokens along the frame (temporal) dimension,

$$\mathbf{h} = \left[\mathbf{h}^{\mathrm{src}}, \mathbf{h}^{\mathrm{tgt}}\right]_{\text{frame-dim}}, \tag{10}$$

and pass $\mathbf{h}$ jointly through the transformer. This allows full attention between the two videos, while each retains its own set of conditioning signals: the source tokens receive only $\mathbf{c}^{\mathrm{src}}$, whereas the target tokens receive $\mathbf{c}^{\mathrm{tgt}}$, $E_{\mathrm{bg}}(\mathbf{z}_{\mathrm{bg}})$, and $E_{\mathrm{mask}}(\mathbf{m})$.

3.3 Dataset for Object-wise Space-Time Retiming

Training and evaluating object-wise retiming requires paired videos with independently controllable object and camera timelines, along with accurate depth, object masks, and camera parameters for correspondence construction. Since such supervision is not readily available in real-world datasets, we construct a large-scale synthetic dataset in Blender. We build diverse scenes using Mixamo (Adobe Systems Inc., 2018) characters and 1,370 motion clips. For each scene, we sample multiple characters with independent motions and render an 81-frame source video at the original object and camera speeds. We then render a paired target with the same geometry, identities, motions, and camera path, while independently retiming each foreground object and the camera.

Each temporal mapping uses either a constant or piecewise-constant playback speed, supporting fast-forward, slow motion, pauses, and reversals. Thus, objects in the same target frame may correspond to different source times, while the camera follows an independent timeline along the same trajectory. We render approximately 16K source-target video pairs with RGB frames, depth maps, object masks, temporal schedules, and per-frame camera poses. These annotations provide the supervision and geometry needed to construct CorrRetime's spatiotemporal correspondences and quantitatively evaluate object-wise and camera retiming. The dataset also includes challenging visibility changes, such as occlusions and newly revealed regions. Additional details are provided in the Appendix.

4 Experiments

4.1 Experimental Settings

Implementation Details. CorrRetime is built on the pretrained Wan2.1-T2V-1.3B model (Wan Team et al., 2025). We train all newly introduced correspondence and conditioning modules, while fully fine-tuning only the self-attention layers of the pretrained backbone and keeping its remaining components frozen. CorrRetime is fine-tuned exclusively on our synthetic dataset. Further implementation details are provided in the Appendix.

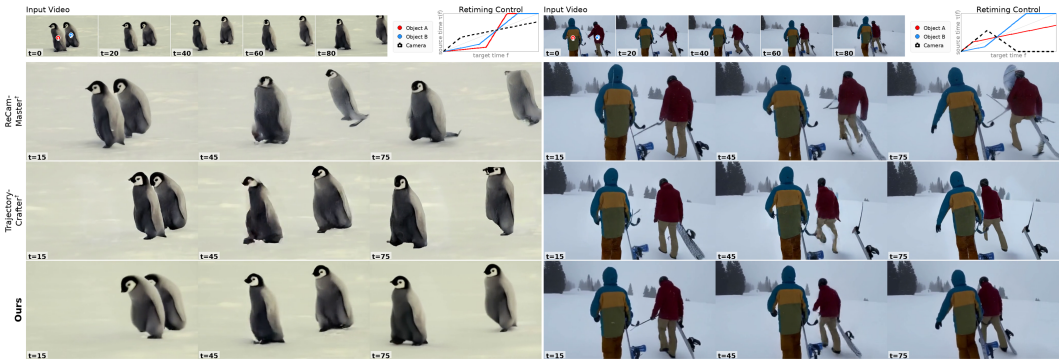

Figure 5: Qualitative comparison on real-world videos. Our method better preserves object identity and geometry under camera-controlled, object-wise retiming, while ReCamMaster† and TrajectoryCrafter† exhibit ghosting, structural artifacts, and inconsistent appearance.

Table 1: Quantitative comparison on our synthetic dataset for object and camera temporal control.

Method	PSNR↑	SSIM↑	LPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓
TrajectoryCrafter	23.738	0.827	0.107	16.034	0.426	0.186
TrajectoryCrafter†	28.702	0.879	0.035	19.042	0.582	0.115
ReCamMaster†	23.352	0.741	0.100	17.383	0.457	0.153
CorrRetime (Ours)	**29.825**	**0.898**	**0.026**	**20.159**	**0.634**	**0.087**

Datasets and Evaluation Metrics. We evaluate object-wise temporal remapping on both synthetic and real-world videos. For synthetic evaluation, we use 386 test sequences with diverse object motions, camera trajectories, and temporal transformations. Since ground-truth retimed videos are available, we report PSNR, SSIM, and LPIPS, along with their masked counterparts mPSNR, mSSIM, and mLPIPS computed over the foreground object regions. For real-world evaluation, we collect 62 videos and generate 1,000 retimed outputs with diverse object-wise and camera schedules. We estimate depth and camera poses using MegaSaM (Li et al., 2025b) and foreground masks using SAM 2 (Ravi et al., 2025). Since ground-truth retimed videos are unavailable, we use an anchor-based protocol in which object states and camera pose coincide with the captured video at a selected anchor frame. We evaluate this frame using PSNR, SSIM, LPIPS, and their masked counterparts, and additionally report FVD for overall video quality. Further details are provided in the Appendix.

4.2 Comparison with State-of-the-Art Methods

Baselines. We adapt TrajectoryCrafter (Yu et al., 2025) and ReCamMaster (Bai et al., 2025) to our object-wise retiming setting, denoted as TrajectoryCrafter† and ReCamMaster†. TrajectoryCrafter† constructs its point-cloud-rendered conditioning video using the same per-object temporal mappings and reprojection geometry as CorrRetime. ReCamMaster† instead uses an object-retimed reference video along the original camera trajectory and performs camera transformation through its native pose conditioning. All adapted baselines use the same Wan2.1-T2V-1.3B backbone and training data as CorrRetime, and are trained for the same number of optimization steps. TrajectoryCrafter without † denotes the original pretrained model. Further details are provided in the Appendix.

Comparison on Synthetic Videos. We evaluate all methods on our synthetic test set with ground-truth retimed videos under varying camera motion. For a fair comparison, the adapted TrajectoryCrafter and ReCamMaster variants are trained on the same data for the same number of optimization steps as CorrRetime. As shown in Table 1, CorrRetime achieves the best performance across all reconstruction and object-region metrics. These results show that directly injecting sparse source-to-target correspondences provides more accurate object-wise retiming than rendered-video conditioning or decoupled camera control.

Comparison on Real-world Videos. We further evaluate CorrRetime on real-world videos using the anchor-based protocol, since ground-truth outputs for arbitrary retiming are unavailable. As shown in Table 2, CorrRetime consistently outperforms the adapted baselines across reconstruction, perceptual, and object-region metrics. It achieves 25.765 PSNR and 0.1041 LPIPS, with 19.112

Table 2: Quantitative evaluation on real-world video retiming.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	mPSNR ↑	mSSIM ↑	mLPIPS ↓	FVD ↓
ReCamMaster†	20.329	0.6252	0.2803	15.762	0.4352	0.2468	526.827
TrajectoryCrafter†	24.426	0.7885	0.1465	17.604	0.5652	0.1750	505.144
CorrRetime (Ours)	**25.765**	**0.8321**	**0.1041**	**19.112**	**0.6110**	**0.1396**	**409.471**

Table 3: **User Study.** We report preference rates (%) for each question across 50 participants, each answering a total of 75 questions.

Method	Q1: Temporal Control	Q2: Subject Consistency	Q3: Overall Quality
ReCamMaster†	18.9	15.8	15.1
TrajectoryCrafter†	20.0	18.2	20.7
CorrRetime (Ours)	**61.1**	**66.1**	**64.2**

Table 4: Ablation of correspondence aggregation, offset encoding, and tuning strategy. Each row differs from the full model in a single component, with all others fixed. Numbers in parentheses denote the difference from the full model.

Ablation	Pooling	Offset	Trainable Modules	mPSNR ↑	mSSIM ↑	mLPIPS ↓
(a) Aggregation	Mean Pool	Yes	Self-Attn	19.483 (-0.545)	0.5802 (-0.0276)	0.0981 (+0.0100)
	Softmax Pool	Yes	Self-Attn	19.743 (-0.285)	0.5934 (-0.0144)	0.0921 (+0.0040)
(b) Offset	Max Pool	No	Self-Attn	19.754 (-0.274)	0.5950 (-0.0128)	0.0953 (+0.0072)
(c) Tuning strategy	Max Pool	Yes	LoRA (rank 64)	18.921 (-1.107)	0.5514 (-0.0564)	0.1079 (+0.0198)
Full	**Max Pool**	**Yes**	**Self-Attn**	**20.028**	**0.6078**	**0.0881**

mPSNR and 0.1396 mLPIPS on foreground regions, and the lowest FVD of 409.471. These results demonstrate improved object-wise retiming accuracy while maintaining overall video quality. Fig. 5 further shows better preservation of object identity and geometry, while the adapted baselines exhibit ghosting, structural artifacts, and appearance inconsistencies. Additional results under diverse object-wise and camera retiming schedules are shown in Fig. 4.

User study. As shown in Table 3, we conduct a user study in which participants compare anonymized, randomly ordered outputs from CorrRetime and competing methods. They evaluate temporal control, subject consistency, and overall quality, selecting their preferred result for each criterion. CorrRetime receives the highest preference across all three criteria.

4.3 Ablation Study

Correspondence aggregation. We investigate how multiple point correspondences assigned to the same latent token should be aggregated. As shown in Table 4(a), max pooling performs best across all masked metrics, outperforming both mean and softmax pooling. This suggests that preserving strong correspondence responses is more effective than averaging them.

Within-token offset encoding. As shown in Table 4(b), removing the within-token offsets consistently degrades all masked metrics. These offsets encode the relative spatial and temporal position of each correspondence within its assigned latent token. While the correspondence identity captures the global source and target coordinates, the offsets preserve fine-grained positional information after tokenization, supporting more accurate object-state transfer.

Fine-tuning strategy. As shown in Table 4(c), we compare full self-attention fine-tuning with a rank-64 LoRA variant. LoRA shows the largest degradation among the tested settings, indicating insufficient capacity to incorporate the correspondence signals. We therefore use full self-attention fine-tuning in the final CorrRetime configuration.

5 Conclusions

We present CorrRetime, a video-to-video framework for fine-grained object-wise temporal retiming with independent camera retiming. Our method establishes explicit spatiotemporal correspondences between each object's source-time state and its target location under the retimed camera trajectory, enabling independent temporal schedules for multiple objects. We further introduce pairwise correspondence conditioning to inject these sparse geometric controls into a pretrained video diffusion model. Across synthetic and real-world evaluations, we demonstrate that CorrRetime improves reconstruction fidelity over adapted baselines.

AI USE STATEMENT

In this work, we used generative AI tools for some tasks requiring disclosure. Specifically, we used them to assist with translation and to provide feedback on the presentation and design of some experiments.

We did not use generative AI tools to generate synthetic datasets, develop the core methodology or conceptual framework, formulate mathematical claims, provide critical ingredients for mathematical proofs, write proofs, propose the main research hypotheses, perform qualitative or thematic data analysis, or generate or manipulate any quantitative or qualitative experimental results. The main ideas, technical contributions, and experimental results presented in this work were developed and produced by the authors.

We also used generative AI tools for tasks for which disclosure is recommended, including editing portions of the manuscript to improve clarity and readability, assisting with the creation and modification of software code, refining the presentation of figures and diagrams, sourcing and searching for information and relevant literature, brainstorming, and suggesting improvements to the structure and presentation of the paper.

All AI-assisted text, code, figures, and other materials were reviewed by the authors. AI-assisted code was manually inspected and tested before use, and all experimental results reported in the paper were independently produced and verified by the authors. We take full responsibility for the final content of this work, including any text, code, figures, claims, or other artifacts produced with the assistance of generative AI.

REFERENCES

Adobe Systems Inc. Mixamo. `https://www.mixamo.com/`, 2018. Accessed: 2026-09-22.

Sherwin Bahmani, Tianchang Shen, Jiawei Ren, Jiahui Huang, Yifeng Jiang, Haithem Turki, Andrea Tagliasacchi, David B. Lindell, Zan Gojcic, Sanja Fidler, Huan Ling, Jun Gao, and Xuanchi Ren. Lyra: Generative 3D scene reconstruction via video diffusion model self-distillation. In *International Conference on Learning Representations (ICLR)*, 2026.

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. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14834–14844. IEEE, 2025.

Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, Yuanzhen Li, Michael Rubinstein, Tomer Michaeli, Oliver Wang, Deqing Sun, Tali Dekel, and Inbar Mosseri. Lumiere: A space-time diffusion model for video generation. In *ACM SIGGRAPH Asia 2024 Conference Papers*, 2024.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023a.

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22563–22575, 2023b.

Ryan Burgert, Charles Herrmann, Forrester Cole, Michael S. Ryoo, Neal Wadhwa, Andrey Voynov, and Nataniel Ruiz. MotionV2V: Editing motion in a video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, Chao Weng, and Ying Shan. VideoCrafter1: Open diffusion models for high-quality video generation. *arXiv preprint arXiv:2310.19512*, 2023.

Daniel Geng, Charles Herrmann, Junhwa Hur, Forrester Cole, Serena Zhang, Tobias Pfaff, Tatiana Lopez-Guevara, Yusuf Aytar, Michael Rubinstein, Chen Sun, Oliver Wang, Andrew Owens, and Deqing Sun. Motion prompting: Controlling video generation with motion trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–12, 2025.

Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, Wenping Wang, and Yuan Liu. Diffusion as shader: 3D-aware video diffusion for versatile video generation control. In *ACM SIGGRAPH 2025 Conference Papers*, 2025. doi: 10.1145/3721238.3730607.

Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. AnimateDiff: Animate your personalized text-to-image diffusion models without specific tuning. In *International Conference on Learning Representations (ICLR)*, 2024.

Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. CameraCtrl: Enabling camera control for video diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen Video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 8633–8646, 2022b.

Zhening Huang, Hyeonho Jeong, Xuelin Chen, Yulia Gryaditskaya, Tuanfeng Y. Wang, Joan Lasenby, and Chun-Hao Huang. SpaceTimePilot: Generative rendering of dynamic scenes across space and time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11218–11228, 2026.

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.

Yao-Chih Lee, Zhoutong Zhang, Jiahui Huang, Jui-Hsien Wang, Joon-Young Lee, Jia-Bin Huang, Eli Shechtman, and Zhengqi Li. Generative video motion editing with 3d point tracks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Quanhao Li, Zhen Xing, Rui Wang, Hui Zhang, Qi Dai, and Zuxuan Wu. MagicMotion: Controllable video generation with dense-to-sparse trajectory guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12112–12123, 2025a.

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)*, pp. 10486–10496. IEEE, 2025b.

Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T. Freeman, and Michael Rubinstein. Layered neural rendering for retiming people in video. *ACM Transactions on Graphics*, 39(6), 2020.

Wan-Duo Kurt Ma, J. P. Lewis, and W. Bastiaan Kleijn. TrailBlazer: Trajectory control for diffusion-based video generation. In *ACM SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024. doi: 10.1145/3680528.3687652.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie Gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloé Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations (ICLR)*, pp. 28085–28128, 2025.

Xiaoyu Shi, Zhaoyang Huang, Fu-Yun Wang, Weikang Bian, Dasong Li, Yi Zhang, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Motion-I2V: Consistent and controllable image-to-video generation with explicit motion modeling. In *ACM SIGGRAPH 2024 Conference Papers*, pp. 1–11, 2024. doi: 10.1145/3641519.3657497.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-video generation without text-video data. In *International Conference on Learning Representations (ICLR)*, 2023.

Wan Team, 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.

Angtian Wang, Haibin Huang, Jacob Zhiyuan Fang, Yiding Yang, and Chongyang Ma. ATI: Any trajectory instruction for controllable video generation. *arXiv preprint arXiv:2505.22944*, 2025a.

Hanlin Wang, Hao Ouyang, Qiuyu Wang, Wen Wang, Ka Leong Cheng, Qifeng Chen, Yujun Shen, and Limin Wang. LeviTor: 3D trajectory oriented image-to-video synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12490–12500, 2025b.

Yiming Wang, Qihang Zhang, Shengqu Cai, Tong Wu, Jan Ackermann, Zhengfei Kuang, Yang Zheng, Frano Rajič, Siyu Tang, and Gordon Wetzstein. BulletTime: Decoupled control of time and camera pose for video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18319–18330, June 2026.

Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. MotionCtrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, pp. 1–11, 2024.

Daniel Watson, Saurabh Saxena, Lala Li, Andrea Tagliasacchi, and David J. Fleet. Controlling space and time with diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

Rundi Wu, Ruiqi Gao, Ben Poole, Alex Trevithick, Changxi Zheng, Jonathan T. Barron, and Aleksander Holynski. CAT4D: Create anything in 4D with multi-view video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26057–26068, 2025.

Weijia Wu, Zhuang Li, Yuchao Gu, Rui Zhao, Yefei He, David Junhao Zhang, Mike Zheng Shou, Yan Li, Tingting Gao, and Di Zhang. DragAnything: Motion control for anything using entity representation. In *European Conference on Computer Vision (ECCV)*, 2024.

Zeqi Xiao, Wenqi Ouyang, Yifan Zhou, Shuai Yang, Lei Yang, Jianlou Si, and Xingang Pan. Trajectory attention for fine-grained video motion control. In *International Conference on Learning Representations (ICLR)*, 2025.

Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. DynamiCrafter: Animating open-domain images with video diffusion priors. In *European Conference on Computer Vision (ECCV)*, 2024.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations (ICLR)*, 2025.

Shengming Yin, Chenfei Wu, Jian Liang, Jie Shi, Houqiang Li, Gong Ming, and Nan Duan. DragNUWA: Fine-grained control in video generation by integrating text, image, and trajectory. *arXiv preprint arXiv:2308.08089*, 2023.

Mark Yu, Wenbo Hu, Jinbo Xing, and Ying Shan. TrajectoryCrafter: Redirecting camera trajectory for monocular videos via diffusion models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 100–111. IEEE, 2025.

Zhenghao Zhang, Junchao Liao, Menghao Li, ZuoZhuo Dai, Bingxue Qiu, Siyu Zhu, Long Qin, and Weizhi Wang. Tora: Trajectory-oriented diffusion transformer for video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2063–2073, 2025.

A Appendix

A.1 Video Demos

We provide supplementary video demos to further illustrate CorrRetime's temporal control capabilities and visual quality. The video first presents qualitative comparisons with adapted TrajectoryCrafter† and ReCamMaster† under identical object-wise and camera retiming settings. We then demonstrate non-zero initial temporal state control, where different objects can start from different source times and subsequently follow independent temporal schedules.

A.2 Qualitative Results with Non-zero Initial Temporal States

Our method does not restrict each object to starting from the first frame of the source video, but allows different objects to begin from different non-zero source times. Thus, within the same target video, individual objects can be initialized from states corresponding to different moments in the source video. From these initial states, each object follows its own temporal schedule, enabling different playback speeds, acceleration, deceleration, pausing, and reversal. As a result, objects can share the same output timeline while exhibiting different initial states and temporal progressions. We demonstrate this additional control capability on real-world videos in Fig. 6 and the demo videos.

A.3 Additional Implementation Details

CorrRetime is built on the pretrained Wan2.1-T2V-1.3B model (Wan Team et al., 2025). To adapt this text-to-video backbone to our video-to-video setting, we encode the source video into the VAE latent space and process it jointly with the noisy target latent: the two latents are patch-embedded separately, concatenated along the temporal dimension, and passed through the diffusion transformer, allowing direct information exchange between source and target representations.

For correspondence conditioning, we sample 512 correspondence points per foreground object for training and all main evaluations. For each object and target frame, we first collect valid pixels within the corresponding source-frame mask and reproject them into the target view using depth and camera geometry. We apply a depth-based z-buffer to retain only the nearest correspondence when multiple source points project to the same target pixel. We then perform target-space stratified sampling over the visible projected correspondences by dividing their target-space bounding box into a 23×23 grid and selecting the correspondence closest to the center of each occupied cell. If more than 512 representatives are obtained, we randomly subsample 512 without replacement; otherwise, we draw additional correspondences from the full set of valid projected correspondences with replacement until reaching 512 points per object. During training, we fine-tune the self-attention layers together with all newly introduced correspondence and conditioning modules, including g_θ, W_{proj}, E_{bg}, and E_{mask}, while keeping all other pretrained components frozen. We train for 13K iterations with a batch size of 8, using AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 10^{-2}). The learning rate starts at 10^{-4} and follows a cosine annealing schedule down to 10^{-5}, without warmup.

Following the flow-matching formulation of the backbone, we sample $\sigma \sim \mathcal{U}(0, 1)$ and construct the noisy latent as

$$\mathbf{x}_\sigma = (1 - \sigma)\, \mathbf{x}_0 + \sigma\, \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{11}$$

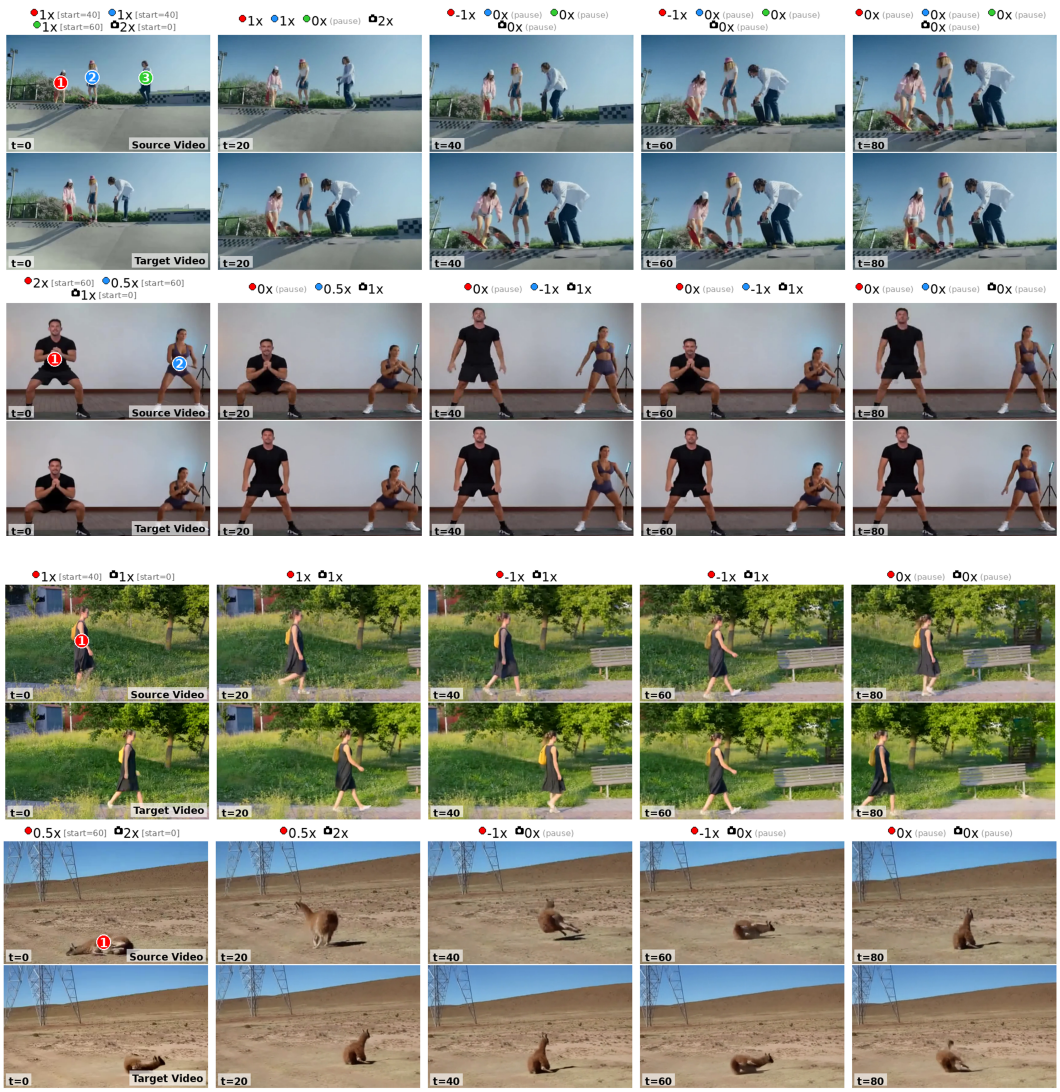

Figure 6: **Non-zero initial temporal state control.** CorrRetime allows different objects to start from different source-time states and subsequently evolve according to independent temporal schedules. Each example shows sampled source frames (top) and the corresponding retimed output (bottom).

The model predicts the velocity $\boldsymbol{\epsilon} - \mathbf{x}_0$ and is trained with a standard mean-squared error loss. All experiments use 81-frame videos at a spatial resolution of 384×672. For the ablation studies, we use a fixed subset of 64 samples from the full set of 386 samples. To ensure a fair comparison, all ablation variants are trained for 9K steps under the same training configuration.

A.4 Effect of Correspondence Point Density

Our model is trained using 512 sampled correspondence points per object. To analyze the sensitivity of CorrRetime to the density of geometric guidance, we keep the trained model fixed and vary only the number of correspondence points used at test time, using $K \in \{32, 64, 128, 256, 512\}$ without retraining or fine-tuning. This allows us to directly evaluate how much correspondence guidance is required at inference time.

As shown in Table 5, increasing the number of correspondence points generally improves reconstruction quality within the retimed object regions, while global reconstruction remains largely stable. mPSNR increases monotonically from 19.305 with 32 points to 20.028 with 512 points, while mLPIPS decreases from 0.1021 to 0.0881. mSSIM similarly increases from 0.5747 to 0.6086 at

Table 5: Ablation on the number of sampled correspondence points per object. We report both global reconstruction metrics (PSNR, SSIM, LPIPS) and object-region metrics (mPSNR, mSSIM, mLPIPS) computed over the retimed foreground regions.

	Global			Object Region		
# Points	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**mPSNR ↑**	**mSSIM ↑**	**mLPIPS ↓**
32	29.245	0.8850	0.0290	19.305	0.5747	0.1021
64	29.099	0.8843	0.0297	19.690	0.5950	0.0942
128	29.315	0.8859	0.0285	19.858	0.6037	0.0910
256	**29.347**	**0.8863**	**0.0281**	19.999	**0.6086**	0.0888
512	29.340	**0.8863**	**0.0281**	**20.028**	0.6078	**0.0881**

256 points and remains nearly unchanged at 512 points. These results indicate that denser correspondence sampling provides more complete spatial guidance for preserving object appearance and geometry during temporal remapping.

In contrast, the global metrics (PSNR, SSIM, LPIPS) vary only marginally across different correspondence densities. This suggests that correspondence density primarily affects the controlled foreground regions, which is consistent with our design: background regions are guided by the dense appearance signal $\mathbf{z}_{\mathrm{bg}}$, whereas retimed foreground objects rely on sparse correspondence conditioning for accurate placement and appearance transfer.

The largest improvements occur between 32 and 128 points, after which the gains gradually saturate, suggesting diminishing returns from denser correspondence guidance. Notably, the model remains robust even with substantially fewer points than used during training, showing only moderate degradation at $K = 32$ ($16\times$ fewer points). This indicates that the correspondence pathway degrades gracefully under sparse test-time guidance. Based on these results, we use 512 correspondence points per object for all final evaluations.

A.5 Additional Baseline Details

TrajectoryCrafter†. TrajectoryCrafter is originally built upon a CogVideoX-based video generation backbone. To train the method directly on our dataset and adapt it to our object-wise retiming setting, we reimplement TrajectoryCrafter by replacing only its backbone with Wan2.1-T2V-1.3B, denoted as TrajectoryCrafter†, while keeping its original dual-stream conditioning mechanism unchanged. This conditioning relies on a point-cloud-rendered reference video and an accompanying mask, which we construct for our object-wise retiming setting as described below; the noisy latent is channel-concatenated with this rendered reference and mask, while the unwarped source video is separately encoded and injected through a dedicated Reference-conditioned Diffusion Transformer (Ref-DiT) block inserted between the backbone's DiT blocks.

To construct this point-cloud-rendered reference for object-wise retiming, we use the same per-object temporal mapping $\tau_k(f)$ and 3D reprojection geometry used by CorrRetime, rather than the single global camera-only reprojection used in the original method. Specifically, for each object O_k and target frame f, we lift the pixels within its segmentation mask at the corresponding source frame $t^{\mathrm{src}}_{k,f}$ into 3D using depth, and reproject them into the target camera view following the same transformation described in Sec. 3.2. Background pixels are independently reprojected using the camera temporal mapping $\tau_{\mathrm{cam}}(f)$. All reprojected points are then composited onto the target frame in depth order, so that a closer object's warped pixels overwrite a farther one's wherever they overlap.

ReCamMaster†. ReCamMaster is originally designed for camera-controlled video generation: given a source video and a target camera trajectory, it synthesizes the scene from the desired viewpoint by concatenating source and target video tokens along the frame dimension and conditioning the resulting sequence on the source and target camera poses through a dedicated camera pose encoding, allowing full spatiotemporal self-attention between the source and target content. However,

ReCamMaster assumes a shared, monotonic timeline for all scene content and therefore does not natively support independently retiming multiple objects.

To adapt ReCamMaster to our object-wise retiming setting, we construct an object-retimed reference video while keeping the camera trajectory of this reference fixed to the original source trajectory. Specifically, for each object O_k and target frame f, we select its appearance from the source frame $t^{\mathrm{src}}_{k,f}$ specified by its temporal mapping $\tau_k(f)$. The selected object pixels are lifted into 3D using depth and reprojected into the *native source-camera viewpoint at frame* f, with inter-object occlusions resolved by depth ordering. The background is kept at its original source-frame appearance. This produces a reference video in which individual objects follow their prescribed temporal schedules, while the reference itself remains aligned with the original camera trajectory.

ReCamMaster† then performs the camera transformation using its original conditioning mechanism. The object-retimed reference is provided as the source video, concatenated with the noisy target tokens along the frame dimension together with the original source camera trajectory P^{src} and the desired retimed target camera trajectory P^{tgt} via its camera pose encoding. Thus, object-wise temporal remapping is introduced through the reference construction, whereas the source-to-target viewpoint transformation is handled by ReCamMaster's native frame-dimension conditioning and camera pose encoding rather than being pre-applied to the reference video.

We denote this adapted variant as ReCamMaster†. For a controlled comparison, we reimplement ReCamMaster on the same Wan2.1-T2V-1.3B backbone used by CorrRetime and TrajectoryCrafter†, initialize its camera-conditioning layers from the official ReCamMaster checkpoint, and train it on the same training data for the same number of optimization steps as our method. This adaptation preserves ReCamMaster's original camera-control mechanism while extending its input construction to support independent object-wise temporal remapping.

Depth and camera pose estimation for synthetic and real-world data. For the synthetic dataset, all models use ground-truth per-frame depth and camera poses rendered from the underlying 3D scenes for both training and evaluation. These geometric inputs are used to construct CorrRetime's correspondences as well as the reprojected reference videos for TrajectoryCrafter† and ReCamMaster†. For real-world evaluation, where ground-truth geometry is unavailable, we estimate depth and camera poses from the source video using MegaSaM (Li et al., 2025b) for all methods. This ensures that all methods use consistent geometric inputs within each evaluation setting.

A.6 Synthetic Dataset Details

We construct our dataset using a Blender-based synthetic environment populated with Mixamo characters and a diverse collection of 1,370 motion clips. For each scene, we randomly select K characters ($K \in \{1, 2, 3\}$), assign each an independent motion clip, and place them at random valid locations within the environment. We first render a *source video*, where all foreground objects and the camera follow the original timeline at $1\times$ speed. From the same scene, we then generate a corresponding *target video* with the same scene geometry, character identities, motions, and camera trajectory, while assigning an independent temporal schedule to each foreground object and the camera. Both source and target videos contain 81 frames and are rendered at a resolution of 1280×720, with the corresponding depth maps and object masks generated at the same spatial resolution. These paired videos are used as training data for CorrRetime. Consequently, the source and target videos share the same underlying scene content and motion assets, but differ in how individual objects and the camera progress over time. Representative examples are shown in Fig. 7.

Temporal schedule. Each temporal schedule is defined as a function $\tau(f)$ that maps a target frame f to a corresponding source time. Rather than relying on a single fixed playback speed, we allow multiple constant-speed segments to be chained together within a single video. Within each segment, an object may move forward at a constant speed, remain stationary, or move backward relative to its previous state. For example, an object may first move at $1\times$, accelerate to $2\times$, pause briefly, and subsequently reverse.

A piecewise schedule consists of 2–4 segments, each spanning at least 15 frames, with speeds sampled from

$$\mathcal{V} = \{-4, -3, -2, -1, -0.5, -\tfrac{1}{3}, -0.25,\ 0,\ 0.25, \tfrac{1}{3}, 0.5, 1, 2, 3, 4\}. \tag{12}$$

The full temporal mapping is obtained by integrating the per-segment speed:

$$\tau(0) = \tau_0, \qquad \tau(f+1) = \tau(f) + s(f), \tag{13}$$

where $s(f)$ denotes the playback speed of the segment containing frame f. Any sampled schedule whose $\tau(f)$ falls outside the valid range of the source timeline is discarded and resampled via rejection sampling. To guarantee that a reverse segment never crosses the starting boundary of the source sequence, we permit a negative speed for a given segment only if the accumulated forward progress at that point, $\tau_0^{\text{seg}} - \tau_0$, is at least as large as the maximum possible displacement the reverse segment could produce, $|s(f)| \times L_{\text{seg}}$, where τ_0^{seg} is the schedule value at the start of the segment and L_{seg} is its length in frames.

Stochastic mixing of constant and piecewise schedules. Piecewise schedules are not applied uniformly to every sample. For each foreground object and for the camera, we independently choose between a constant-speed schedule and a piecewise schedule, applying the piecewise variant with probability 0.5 in our implementation. As a result, within a single clip, some objects may follow a constant-speed schedule while others follow a piecewise schedule, and the camera is retimed independently of all objects.

Object-wise independence. Crucially, each object is assigned an independent temporal mapping. The target-frame state of object O_k can be conceptually written as

$$O_k^{\text{tgt}}(f) = O_k^{\text{src}}\big(\tau_k(f)\big). \tag{14}$$

Thus, at the same target frame f, different objects may correspond to entirely different source times, as expressed in Eq. 14. For instance, one object may correspond to source frame 60, a second to frame 30, and a third to frame 45, all within the same target frame.

Object-wise retiming in Blender with Non-Linear Animation (NLA). In Blender, we implement object-wise retiming using one NLA strip per temporal segment. Each strip maps a contiguous range of scene (output) frames to a corresponding range of action (source-animation) frames, with the mapping slope determined by the segment speed. Positive-speed segments play the action forward, whereas negative-speed segments reverse the playback direction so that action time decreases while scene time continues to advance. Zero-speed segments are handled within the same mechanism by collapsing the action-frame range to a small interval of width ϵ around the segment's starting pose and stretching this near-zero range over the full scene-frame duration. This effectively holds the character at a fixed pose without requiring a separate freeze operation.

A key implementation detail is the treatment of segment boundaries. For segment i, spanning scene frames $[\text{start}_i, \text{end}_i]$, we set its ending action-frame position to $\tau(\text{end}_i + 1)$ rather than $\tau(\text{end}_i)$, and we set the starting action-frame position of the following segment $i+1$ to the same value, $\tau(\text{start}_{i+1})$. For example, if segment i ends at scene frame 30, its action-frame endpoint is set to $\tau(31)$, and segment $i+1$, starting at scene frame 31, begins from that identical value $\tau(31)$. Since consecutive segments satisfy $\text{end}_i + 1 = \text{start}_{i+1}$, adjacent NLA strips meet at exactly the same action-frame coordinate. This avoids the one-frame pose discontinuity that arises when each strip independently derives its source range only from its own $[\text{start}_i, \text{end}_i]$ interval. For the final segment, where no subsequent boundary exists, we extrapolate the action-frame endpoint by one additional step using the segment's current speed, thereby preserving the same per-frame temporal progression as in the rest of the schedule.

Camera retiming. The camera is assigned its own temporal schedule independently of the foreground objects. Rather than generating a new camera trajectory for the target video, we preserve the same spatial camera path used in the source video and only change how the camera progresses along that path over time. Specifically, each target frame uses the camera position and orientation corresponding to the source time specified by $\tau_{\text{cam}}(f)$. This allows the camera to speed up, slow down, pause, or move backward along the same orbit, dolly, or truck trajectory. To ensure that the source and target videos follow exactly the same spatial path, we reuse the same randomly sampled camera-motion parameters when rendering both videos. We additionally compute and store the camera pose at every frame, rather than relying on sparse keyframes and interpolation, so that piecewise changes such as pauses and reversals are represented accurately.

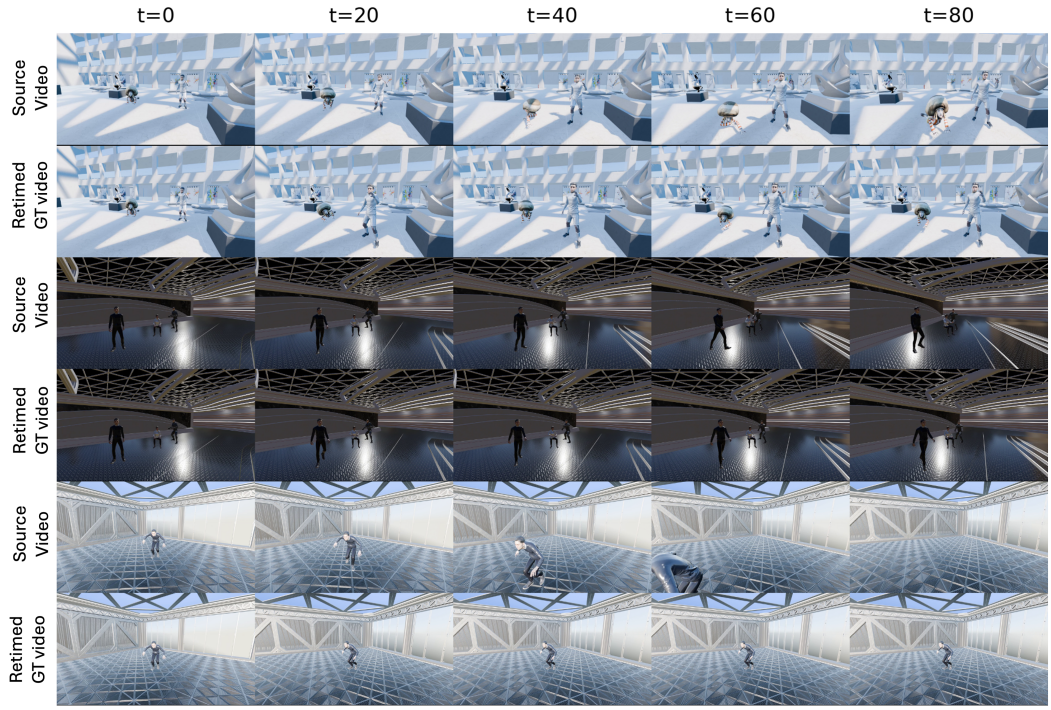

Figure 7: Examples from our synthetic dataset. Each pair shows a source video and its corresponding retimed target video, where individual objects and the camera are independently retimed according to their assigned temporal schedules.

Overall formulation. A single target video is thus generated with a set of independent time axes,

$$\{\tau_1(f), \tau_2(f), \ldots, \tau_K(f), \tau_{\text{cam}}(f)\}. \tag{15}$$

For example, at a given target frame, if

$$\tau_1(f) = 65, \qquad \tau_2(f) = 30, \qquad \tau_{\text{cam}}(f) = 45, \tag{16}$$

then the first object corresponds to the 65-th source time step, the second object to the 30-th, while the background and viewpoint correspond to the 45-th time step of the source camera trajectory, consistent with the set of independent schedules in Eq. equation 15.

Stored data and metadata. For every rendered scene, we save the following data and metadata to disk:

- **Rendered RGB videos** — the normal source video, in which all foreground objects and the camera evolve at constant $1\times$ speed, and the corresponding GT video generated under the independently sampled object-wise and camera temporal schedules described above.
- **`temporal_schedule.json`** — the exact temporal schedule used for the GT render, including the per-frame $\tau(f)$, per-frame velocity, and the start/end frame and speed of every segment, recorded separately for each foreground object and for the camera. This file serves as the single source of truth for temporal mapping: correspondence computation and reference-video generation read the stored schedule directly rather than resampling a new one, ensuring that the GT video and its conditioning/reference share exactly the same temporal mapping.
- **`camera.json`** — per-frame camera geometry, including the 3D world position, Euler-angle rotation, and the full 4×4 camera-to-world transformation matrix. We additionally store fixed camera parameters such as focal length, sensor width, horizontal and vertical field-of-view, image resolution, and intrinsic parameters (f_x, f_y, c_x, c_y).

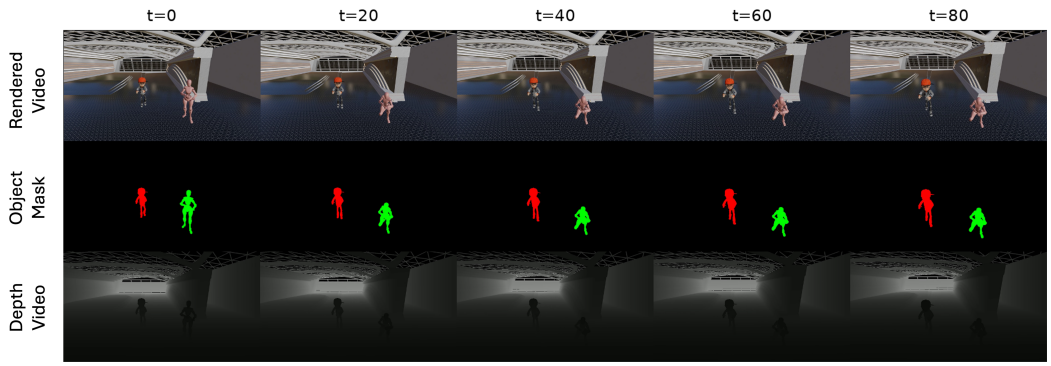

Figure 8: Examples of generated samples from our synthetic dataset. Each sequence includes rendered RGB frames, per-object segmentation masks, and corresponding depth maps.

- **`bbox.json`** — per-frame, per-character 2D bounding boxes in both normalized and pixel coordinates, computed by projecting the character mesh vertices into the camera view. We also store each character's 3D world-space root position for every frame.
- **`characters.json`** — static per-scene metadata for each character, including the character asset, motion clip, initial 3D position, projected 2D screen coordinates of the head/center/foot, and a human-readable summary of the average playback speed. Fine-grained per-frame temporal information is stored exclusively in `temporal_schedule.json`.
- **Depth video** — a per-pixel camera-space depth pass rendered using a material-override emission shader that encodes normalized linear Z-depth, clipped to a fixed maximum depth. We export an 8-bit visualization (`depth_viz.mp4`, H.264) for quick inspection and a 16-bit lossless version (`depth.mkv`, FFV1) for downstream geometric processing.
- **Object mask video** — a per-pixel character instance-segmentation pass rendered using a material-override shader that encodes each character's object index as a constant emission value. We export a lossless single-channel mask (`objmask.mkv`, FFV1) for downstream processing, together with an 8-bit color-coded visualization (`objmask_viz.mp4`) for human inspection.

For the depth and object-mask passes, anti-aliasing, ambient occlusion, screen-space reflections, and bloom are disabled, and the view transform is set to `Raw` without tone mapping. This ensures that the exported pixel values preserve the underlying linear depth and object-index values rather than being altered by the standard RGB rendering pipeline. Examples of the generated data are shown in Fig. 8.

A.7 Additional Details on Anchor-Based Evaluation for Real-World Videos

Motivation. Unlike our synthetic dataset, real-world videos do not provide ground-truth retimed targets. In the synthetic setting, both the source video and its retimed target can be rendered from the same underlying 3D scene, enabling direct frame-wise comparison. In contrast, for a real-world video, only the original recording is available, and there is no physically observed ground-truth video corresponding to independently retimed objects and camera motion. Evaluating arbitrary retimed frames against a synthesized pseudo target would therefore conflate model error with inaccuracies introduced by the pseudo-ground-truth construction itself.

To enable quantitative evaluation, we construct *anchor-constrained temporal schedules*. The key idea is to allow each object and the camera to follow a non-trivial retiming schedule while forcing all temporal mappings to coincide with the original timeline at a common anchor frame a:

$$\tau_k(a) = a, \qquad \tau_{\text{cam}}(a) = a, \tag{17}$$

where τ_k denotes the temporal mapping of object k and τ_{cam} denotes that of the camera. Consequently, although the scene is temporally rearranged away from the anchor, the original video frame at a provides an exact reference at the anchor itself.

Anchor-Constrained Schedule. For a desired playback speed s, we construct a two-segment temporal schedule satisfying both $\tau(0) = 0$ and $\tau(a) = a$. We use different constructions for positive and negative playback speeds.

For $s > 0$ and $s \neq 1$, the first segment uses a compensating speed

$$s_{\text{comp}} = \frac{1}{s}, \tag{18}$$

while the second segment uses the target speed s. Let b denote the breakpoint between the two segments. We choose b such that the accumulated source time returns exactly to the original timeline at the anchor:

$$s_{\text{comp}} b + s(a - b) = a. \tag{19}$$

Solving for b gives

$$b = a\frac{1 - s}{s_{\text{comp}} - s} = \frac{as}{1 + s}, \tag{20}$$

and the resulting mapping is

$$\tau(f) = \text{clip}\left(\begin{cases} s_{\text{comp}} f, & f \leq b, \\ s_{\text{comp}} b + s(f - b), & f > b, \end{cases}; [0, F - 1]\right). \tag{21}$$

For $s = 1$, we use the identity schedule $\tau(f) = f$.

For a requested negative speed $s < 0$, we use a separate construction. We first advance forward at a fixed speed $s_{\text{fwd}} = 2$, and then reverse toward the anchor. We compute a continuous breakpoint

$$b^* = a\frac{1 - s}{s_{\text{fwd}} - s}, \qquad s_{\text{fwd}} = 2, \tag{22}$$

and use the integer breakpoint

$$b = \lfloor b^* \rfloor. \tag{23}$$

Because discretizing the breakpoint can slightly change the temporal displacement, we recompute the reverse-segment speed as

$$s_{\text{rev}} = \frac{a - s_{\text{fwd}} b}{a - b}, \tag{24}$$

which ensures that $\tau(a) = a$ remains exact. The negative-speed schedule is therefore

$$\tau(f) = \text{clip}\left(\begin{cases} s_{\text{fwd}} f, & f \leq b, \\ s_{\text{fwd}} b + s_{\text{rev}}(f - b), & f > b, \end{cases}; [0, F - 1]\right). \tag{25}$$

Thus, the motion first advances beyond the source time associated with the anchor and then reverses to recover the original state exactly at frame a. For both positive and negative schedules, the second segment continues beyond the anchor for the remainder of the video, and source times outside the valid range are clipped to $[0, F - 1]$.

For example, for an 81-frame video ($F = 81$) with anchor $a = 60$ and positive target speed $s = 2$, we obtain $s_{\text{comp}} = 0.5$ and $b = 40$. The temporal mapping therefore satisfies

$$\tau(20) = 10, \qquad \tau(40) = 20, \qquad \tau(50) = 40, \qquad \tau(60) = 60. \tag{26}$$

The motion first progresses more slowly than the original video and then accelerates after the breakpoint, exactly recovering the original source time at the anchor. Fig. 9 visualizes representative anchor-constrained schedules for different playback speeds and anchor locations.

Independent Object and Camera Schedules. The anchor-constrained schedule is constructed independently for each foreground object and for the camera. Thus, different objects may follow different target speeds and have different breakpoints, while all schedules share the same anchor frame. For example, the camera may use $2\times$ playback, while different objects use $4\times$, $0.5\times$, or $1\times$ playback. Their temporal trajectories differ over most of the sequence, but they are constrained to satisfy

$$\tau_{\text{cam}}(a) = \tau_1(a) = \cdots = \tau_K(a) = a. \tag{27}$$

Therefore, at the anchor frame, the camera, background, and all foreground objects simultaneously recover the same temporal state as in the original video.

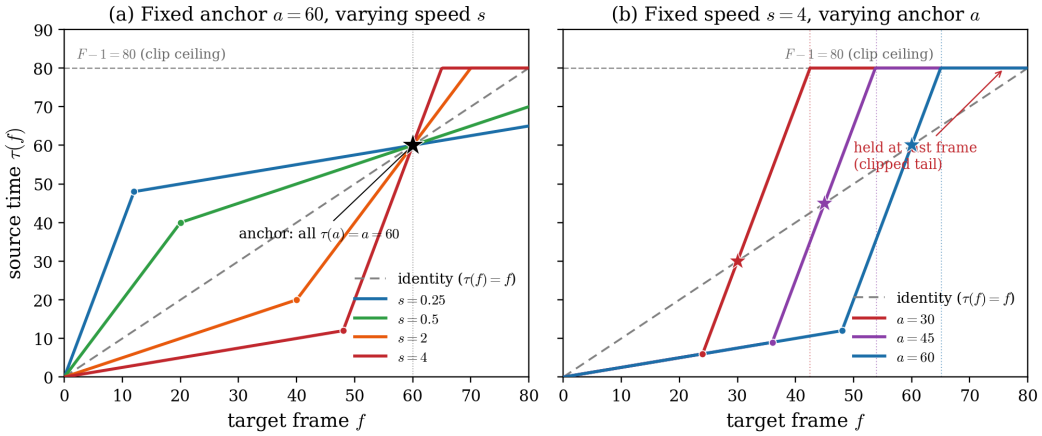

Figure 9: Visualization of the anchor-constrained temporal schedules used for real-world evaluation. (a) With a fixed anchor at $a = 60$, different target speeds produce different piecewise-linear temporal mappings, while all schedules reconverge to the original timeline at the anchor, i.e., $\tau(a) = a$. (b) With a fixed target speed of $s = 4$, varying the anchor location changes the breakpoint while preserving the anchor constraint. Source times outside the valid range are clipped to $[0, F - 1]$.

Evaluation at the Anchor Frame. For non-anchor frames, different scene components generally correspond to different source times. Such retimed configurations do not have an exact real-world ground-truth counterpart, and evaluating them using pixel-level metrics would require an approximate pseudo target. We therefore restrict quantitative evaluation to the anchor frame, where all temporal mappings are guaranteed to coincide with the original timeline.

At the anchor, the original frame can be used directly as the ground truth. We compute PSNR, SSIM, and LPIPS over the full frame, together with mPSNR, mSSIM, and mLPIPS over the foreground object regions. This allows quantitative evaluation using an exact observed reference rather than an approximate synthesized target.

Anchor Selection. We sample requested target speeds from

$$s \in \{-4,\ -3,\ -2,\ -1,\ -0.5,\ 0.25,\ 0.5,\ 1,\ 2,\ 4\} \tag{28}$$

and choose anchor locations from

$$a \in \{30,\ 45,\ 60\}. \tag{29}$$

For positive non-unit speeds, these anchor positions yield integer breakpoints under Eq. equation 20, avoiding fractional-frame rounding. For example, when $a = 60$,

$$\begin{aligned} s = 0.25 &\Rightarrow b = 12, \\ s = 0.5 &\Rightarrow b = 20, \\ s = 2 &\Rightarrow b = 40, \\ s = 4 &\Rightarrow b = 48. \end{aligned} \tag{30}$$

For negative speeds, we instead floor the continuous breakpoint to an integer frame and recompute the reverse-segment speed as described above, ensuring that $\tau(a) = a$ remains exact. Fig. 9(b) further illustrates how changing the anchor location shifts the breakpoint while preserving the anchor constraint.